Project-Team RMoD 2013 Activity Report

Marcus Denker, Nicolas Anquetil, Damien Cassou, Stéphane Ducasse, Anne Etien, Damien Pollet

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

Activity Report 2013

Project-Team RMOD

Analyses and Languages Constructs for Object-Oriented Application Evolution

IN COLLABORATION WITH: Laboratoire d'informatique fondamentale de Lille (LIFL)

RESEARCH CENTER
Lille - Nord Europe

THEME
Distributed programming and software engineering

Table of contents

Project-Team RMOD

Keywords: Software Engineering, Software Evolution, Maintenance, Reflective Programming Languages, Dynamic Languages

Creation of the Project-Team: 2009 July 01.

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]
Damien Cassou [Univ. Lille I, Associate Professor]
Anne Etien [Univ. Lille I, Associate Professor]
Damien Pollet [Univ. Lille I, Associate Professor]

External Collaborator
Olivier Auverlot [Univ. Lille I, Engineer]

Engineers
Clément Bera [Inria]
Muhammad Bhatti [Inria, granted by OSEO Innovation]
Guillaume Larcheveque [Inria]
Esteban Lorenzano [Inria, granted by OSEO Innovation]
Nicolas Petton [Inria, granted by OSEO Innovation]
Igor Stasenko [Inria]
Simon Allier [Inria, util Apr 2013]

PhD Students
Camillo Bruni [Inria]
Andre Cavalcante Hora [Inria, granted by ANR CUTTER project]
Martin Dias [Inria, granted by REGION NORD PAS DE CALAIS]
Nikolaos Papoulias [Ecole des Mines de Douai, co-supervision, until Sep 2013]
Guillermo Polito [Ecole des Mines de Douai, co-supervision]
Camille Teruel [Inria]

Post-Doctoral Fellows
Jean-Baptiste Arnaud [Inria, granted by OSEO Innovation]
Fernando Olivero [Inria, Aug/Sep 2013]

Visiting Scientist
Rafael Serapilha Durelli [Visiting PhD Student, University of São Paulo, Brazil, from Feb 2013]

Administrative Assistants
Christelle Gasperini [Inria, until Oct 2013]
Julie Jonas [Inria, from Oct 2013]

Others
Benjamin Van Ryseghem [Student, Univ. Lille I]
Benjamin Arezki [Student, Univ. Lille I, until Jan 2013]
Vincent Blondeau [Student, Univ. Lille I, from May until Aug 2013]
Gustavo Jansen de Souza Santos [Intern, Federal University of Minas Gerais, Brasil, from Sep until Nov 2013]
Gisela Decuzzi [Intern, Universidad Tecnológica Nacional FRBA, Argentina, from Apr until Jul 2013]

Mathieu Dehouck [Student, Univ. Lille I, from Apr until Jun 2013]
Erwan Douaille [Intern, Univ. Lille I, until Jun 2013]
Pablo Herrero [Intern, Universidad de Buenos Aires, Argentina, from Sep 2013]
Kevin Lanvin [Student, Univ. Lille I, from Dec 2013]
Sebastian Tleye [Intern, Universidad de Buenos Aires, Argentina, from Apr until Sep 2013]
Yuriy Tymchuk [Intern, Ivan Franko National University of Lviv, Ukraine, from Jan until Apr 2013]

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 [64]. 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 [61]. 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 [70]. 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.

2.4. Highlights of the Year

- Stéphane Ducasse got promoted DR1 (December 2012).
- A *Web with Pharo* Conference was held 6 June 2013 @ Euratechnologies, Lille
- Pharo 2.0 (our open-source language and environment) was released. (http://www.pharo.org)
- Three releases of Moose: 4.7, 4.8 and 4.9. Moose is our open-source reengineering platform. (http://www.moosetechnology.org)
- The second PharoConf was held at University of Bern, Switzerland April 2-4.
- The first ever Pharo Tutorial at ECOOP in 2013.
- RMoD helped to Organize the Dyla workshop at ECOOP 2013.
- Creation of Synectique. The company is a spin-off based on the research done in RMoD. Synectique is selling software maintenance solutions based on Pharo. (http://www.synectique.eu)
- RMoD participated to the organization of the ESUG conference in Annecy, France in September (over 100 participants).
- *Deep into Pharo* Book Released. *Deep into Pharo* is the second volume of a series of books covering Pharo. (http://rmod.lille.inria.fr/deepIntoPharo/)
- Organization of the MooseDay in Lille on the 19th December with around 25 persons from all around the world.

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 [55], [54]. 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) [83] 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 [42]. 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 [71]. Until now, few works have attempted to identify layers in practice: Mudpie [85] is a first cut at identifying cycles between packages as well as package groups potentially representing layers. DSM (dependency structure matrix) [84], [79] 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 [67]. 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 [86], [58].

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 apparition 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 [81], [56] and classboxes [43] 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 [81], [56]. 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 [53], [75], [44], [57] and several type systems were defined [59], [82], [76], [69].

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 [46]. 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 [56], stateful [45], and freezable [57]) 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 [74]. 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 [51] then from Smalltalk [62].

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 [66]). 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 [47], as well as controlling the access to reflective features [48], [49] are important challenges. We plan to:

- Study the isolation abstractions available in erights (http://www.erights.org) [73], [72], and Java's class loader strategies [68], [63].
- Categorize the different reflective features of languages such as CLOS [65], Python and Smalltalk [77] and identify suitable isolation mechanisms and infrastructure [60].
- Assess different isolation models (access rights, capabilities [78]...) 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 [47],
  - the use encapsulation policies as a basis to restrict the interfaces of the controlled objects [80],
  - 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...) [77].

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 over 10 company members).

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. Software and Platforms

5.1. Moose

Participants: Stéphane Ducasse [correspondant], Muhammad Bhatti, Andre Calvante Hora, Nicolas Anquetil, Anne Etien, Guillaume Larcheveque, Tudor Gîrba [University of Bern].

Web: http://www.moosetechnology.org/

The platform. 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, a model repository, and generic GUI support for querying, browsing and grouping. The development of Moose began at the Software Composition Group in 1997, and is currently contributed to and used by researchers in at least seven European universities. Moose offers an extensible meta-described metamodel, a query engine, a metric engine and several visualizations. Moose is currently in its fourth major release and comprises 55,000 lines of code in 700 classes.

The RMoD team is currently the main maintainer of the Moose platform. There are 200 publications (journal, international conferences, PhD theses) based on execution or use of the Moose environment.

The first version running on top of Pharo (Moose 4.0) was released in June 2010. In 2013, three releases of Moose where done: 4.7 to 4.9. The current focus is Moose 5.0, which is running on Pharo3 and will be released together with Pharo3 in spring 2014.
Here is the self-assessment of the team effort following the grid given at http://www.inria.fr/institut/organisation/instances/commission-d-evaluation.

- **(A5)** Audience : 5 – Moose is used by several research groups, a consulting company, and some companies using it in ad-hoc ways.
- **(SO4)** Software originality : 4 – Moose aggregates the last results of several research groups.
- **(SM4)** Software Maturity : 4 – Moose is developed since 1996 and got two main redesign phases.
- **(EM4)** Evolution and Maintenance : 4 – Moose will be used as a foundation of our Synectique start up so its maintenance is planned.
- **(SDL4)** Software Distribution and Licensing : 4 – Moose is licensed under BSD
- **(OC)** Own Contribution : (Design/Architecture)DA-4, (Coding/Debugging)-4, (Maintenance/Support)-4, (Team/Project Management)-4

5.2. Pharo

Participants: Marcus Denker [correspondant], Damien Cassou, Stéphane Ducasse, Esteban Lorenzano, Damien Pollet, Igor Stasenko, Camillo Bruni, Camille Teruel, Clément Bera.

Web: http://www.pharo.org/

The platform. Pharo is a new 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.

The first stable version, Pharo 1.0, was released in 2010. The development of Pharo accelerated in 2011 and 2012: Versions 1.2 to 1.4 have been released (with more than 2400 closed issues). In 2013, Pharo 2.0 was released. The development cycle will now be one major release per year, with Pharo3 to be released in March 2014.

In 2012, RMoD organized the first *Pharo Conference* during two days in May with 60 participants, the second Pharo conference was held in Bern, Switzerland in 2013.

Additionally, in November 2012 RMoD launched the Pharo Consortium (http://consortium.pharo.org/) and the Pharo Association (http://association.pharo.org). Over 10 companies are now paying members of the Consortium.

RMoD is the main maintainer and coordinator of Pharo.
Here is the self-assessment of the team effort following the grid given at http://www.inria.fr/institut/organisation/instances/commission-d-evaluation.

- **(A5)** Audience: 5 – Used in many universities for teaching, more than 25 companies.
- **(SO3)** Software originality : 3 – Pharo offers a classical basis for some aspects (UI). It includes new frameworks and concepts compared to other Smalltalk implementations.
- **(SM4)** Software Maturity: 4 – Bug tracker, continuous integration, large test suites are on place.
- **(EM4)** Evolution and Maintenance: 4 – Active user group, consortium and association had just been set up.
- **(SDL4)** Software Distribution and Licensing: 4 – Pharo is licensed under MIT.
- **(OC5)** Own Contribution: (Design/Architecture) DA-5, (Coding/Debugging) CD-5, (Maintenance/Support) MS-5, (Team/Project Management) TPM-5

5.3. Fuel

Participants: Martin Dias [Correspondant], Mariano Martinez-Peck.

Web: http://rmod.lille.inria.fr/web/pier/software/fuel

Objects in a running environment are constantly being born, mutating their status and dying in the volatile memory of the system. The goal of serializers is to store and load objects either in the original environment or in another one. Fuel is a general-purpose serializer based on four principles: (1) speed, through a compact binary format and a pickling algorithm which obtains the best performance on materialization; (2) good object-oriented design, without any special help from the virtual machine; (3) specialized for Pharo, so that core objects (such as contexts, block closures and classes) can be serialized too; (4) flexible about how to serialize each object, so that objects are serialized differently depending on the context.

Since Pharo 2.0, Fuel is part of the standard distribution.
Here is the self-assessment of the team effort following the grid given at http://www.inria.fr/institut/organisation/instances/commission-d-evaluation.

- **(A4)** Audience: 4 – Large audience software, usable by people inside and outside the field with a clear and strong dissemination, validation, and support action plan.
- **(SO3)** Software originality : 3.
- **(SM4)** Software Maturity: 4 – Bug tracker, continuous integration, large test suites are on place.
- **(EM4)** Evolution and Maintenance: 4.
- **(SDL4)** Software Distribution and Licensing: 4 – Fuel is licensed under MIT.
- **(OC5)** Own Contribution: (Design/Architecture) DA-5, (Coding/Debugging) CD-5, (Maintenance/Support) MS-5, (Team/Project Management) TPM-5

5.4. Athens

Participant: Igor Stasenko [Correspondant].

Athens is a vector graphics framework for Pharo. Athens is now part of Pharo since version 3 as a technology Preview. We plan to make Athens the default graphics framework with Pharo4 in 2015.

5.5. Citezen

Participants: Damien Pollet [Correspondant], Stéphane Ducasse.

Web: http://people.untyped.org/damien.pollet/software/citezen/

Citezen is a suite of tools for parsing, validating, sorting and displaying BibTeX databases. This tool suite is integrated within the Pier Content Management System (CMS) and both are implemented on top of Pharo. Citezen aims at replacing and extending BibTeX, in Smalltalk; ideally, features would be similar to BibTeX, CrossTeX, and CSL.

5.6. Handles

Participant: Jean-Baptiste Arnaud [Correspondant].

Web: http://jeanbaptiste-arnaud.eu/handles/

An Handle is a first-class reference to a target object. Handles can alter the behavior and isolate the state of the target object. Handles provide infrastructure to automatically create and wrap new handles when required. A real-time control of handles is possible using a special object called metaHandle.

5.7. Hazelnut

Participants: Guillermo Polito [Correspondant], Benjamin Van Ryseghem, Nicolas Paez, Igor Stasenko.

Web: http://rmod.lille.inria.fr/web/pier/software/Seed

Traditionally, Smalltalk-based systems are not bootstrapped because of their ability to evolve by self-modification. Nevertheless, the absence of a bootstrap process exposes many problems in these systems, such as the lack of reproducibility and the impossibility to reach certain evolution paths. Hazelnut is a tool that aims to introduce a bootstrap process into these systems, in particular Pharo.

5.8. LegacyParsers

Participants: Muhammad Bhatti [Correspondant], Nicolas Anquetil, Guillaume Larcheveque, Esteban Lorenzano, Gogui Ndong.

As part of our research on legacy software and also for the Synectique company), we started to define several parsers for old languages like Cobol for example. This work is important to help us validate our meta-model and tools against a larger range of existing technologies and to discover the limits of our approach. From our initial results, and the in-depth understanding that it gave us, we are formulating new research objectives in meta-model driven reverse engineering. This work is also important for the spin-off company, as being able to work with such technologies is fundamental.

5.9. Mate

Participants: Marcus Denker [Correspondant], Clement Bera, Camillo Bruni.

Mate is the future research-oriented virtual machine for Pharo. Its goal is to serve as a prototype for researchers to experiment with. As a result, the design of Mate is very simple to understand. As of today, Mate consists of an AST interpreter, a new object memory layout, and a simple garbage collector.

5.10. NativeBoost

Participant: Igor Stasenko [Correspondant].

Web: http://code.google.com/p/nativeboost/

NativeBoost is a Smalltalk framework for generating and running machine code from the language side of Pharo. As part of it comes a foreign function interface that enables calling external C functions from Smalltalk code with minimal effort.

5.11. Nabujito

Participants: Camillo Bruni [Correspondant], Marcus Denker.

Nabujito is an experimental Just In Time compiler implemented as a Smalltalk application, based on NativeBoost, that does not require changes in the virtual machine.

5.12. Nautilus

Participants: Benjamin Van Ryseghem [Correspondant], Stéphane Ducasse, Igor Stasenko, Camillo Bruni, Esteban Lorenzano.

Nautilus is a new source code browser based on the latest infrastructure representations. Its goal is mainly to replace the current system browser that was implemented in the 80s and that doesn't provide optimal tools for the system as it has evolved.

5.13. Spec

Participants: Benjamin Van Ryseghem [Correspondant], Stéphane Ducasse, Johan Fabry.

Spec is a programming framework for generating graphical user interfaces inspired by VisualWorks' Subcanvas. The goal of Spec is to tackle the lack of reuse experienced in existing tools. Spec serves as a pluggable layer on top of multiple lower-level graphical frameworks. Many improvements have been noticed in Pharo after the introduction of Spec in terms of speed or number of lines of code while we re-implemented existing tools using Spec.

6. New Results

6.1. Tools for understanding applications: IDEs and Visualization

Performance Evolution Blueprint: Understanding the Impact of Software Evolution on Performance. Understanding the root of a performance drop or improvement requires analyzing different program executions at a fine grain level. Such an analysis involves dedicated profiling and representation techniques. JProfiler and YourKit, two recognized code profilers fail, on both providing adequate metrics and visual representations, conveying a false sense of the performance variation root. We propose performance evolution blueprint, a visual support to precisely compare multiple software executions. Our blueprint is offered by Rizel, a code profiler to efficiently explore performance of a set of benchmarks against multiple software revisions. [31]

Seamless Composition and Reuse of Customizable User Interfaces with Spec Implementing UIs is often a tedious task. To address this, UI Builders have been proposed to support the description of widgets, their location, and their logic. A missing aspect of UI Builders is however the ability to reuse and compose widget logic. In our experience, this leads to a significant amount of duplication in UI code. To address this issue, we built Spec: a UIBuilder for Pharo with a focus on reuse. With Spec, widget properties are defined declaratively and attached to specific classes known as composable classes. A composable class defines its own widget description as well as the model-widget bridge and widget interaction logic. Spec enables seamless reuse of widgets, its use in Pharo 2.0 has cut in half the amount of lines of code of six of its tools, mostly through reuse. This shows that Spec meets its goals of allowing reuse and composition of widget logic. [17]

Pragmatic Visualizations for Roassal: a Florilegium Software analysis and in particular reverse engineering often involves a large amount of structured data. This data should be presented in a meaningful form so that it can be used to improve software artefacts. The software analysis community has produced numerous visual tools to help understand different software elements. However, most of the visualization techniques, when applied to software elements, produce results that are difficult to interpret and comprehend. We present five graph layouts that are both expressive for polymetric views and agnostic to the visualization engine. These layouts favor spatial space reduction while emphasizing on clarity. Our layouts have been implemented in the Roassal visualization engine and are available under the MIT License. [23]

6.2. Software Quality: Bugs and Debuggers

BugMaps-Granger: A Tool for Causality Analysis between Source Code Metrics and Bugs. Despite the increasing number of bug analysis tools for exploring bugs in software systems, there are no tools supporting the investigation of causality relationships between internal quality metrics and bugs. We propose an extension of the BugMaps tool called BugMaps-Granger that allows the analysis of source code properties that caused bugs. For this purpose, we relied on Granger Causality Test to evaluate whether past changes to a given time series of source code metrics can be used to forecast changes in a time series of defects. Our tool extracts source code versions from version control platforms, generates source code metrics and defects time series, computes Granger, and provides interactive visualizations for causal analysis of bugs. We also provide a case study in order to evaluate the tool. [22]

Mining Architectural Patterns Using Association Rules

Software systems usually follow many programming rules prescribed in an architectural model. However, developers frequently violate these rules, introducing architectural drifts in the source code. We present a data mining approach for architecture conformance based on a combination of static and historical software analysis. For this purpose, the proposed approach relies on data mining techniques to extract structural and historical architectural patterns. In addition, we propose a methodology that uses the extracted patterns to detect both absences and divergences in source-code based architectures. We applied the proposed approach in an industrial strength system. As a result we detected 137 architectural violations, with an overall precision of 41.02%. [27]

Heuristics for Discovering Architectural Violations

Software architecture conformance is a key software quality control activity that aims to reveal the progressive gap normally observed between concrete and planned software architecture. We present ArchLint, a lightweight approach for architecture conformance based on a combination of static and historical source code analysis. For this purpose, ArchLint relies on four heuristics for detecting both absences and divergences in source code based architectures. We applied ArchLint in an industrial-strength system and as a result we detected 119 architectural violations, with an overall precision of 46.7% and a recall of 96.2%, for divergences. We also evaluated ArchLint with four open-source systems, used in an independent study on reflexion models. In this second study, ArchLint achieved precision results ranging from 57.1% to 89.4%. [26]

6.3. Software Quality: History and Changes

Representing Code History with Development Environment Events. Modern development environments handle information about the intent of the programmer: for example, they use abstract syntax trees for providing high-level code manipulation such as refactorings; nevertheless, they do not keep track of this information in a way that would simplify code sharing and change understanding. In most Smalltalk systems, source code modifications are immediately registered in a transaction log often called a ChangeSet. Such mechanism has proven reliability, but it has several limitations. We analyse such limitations and describe scenarios and requirements for tracking fine-grained code history with a semantic representation. We want to enrich code sharing with extra information from the IDE, which will help understanding the intention of the changes and let a new generation of tools act in consequence. [24]

Mining System Specific Rules from Change Patterns A significant percentage of warnings reported by tools to detect coding standard violations are false positives. Thus, there are some works dedicated to provide better rules by mining them from source code history, analyzing bug-fixes or changes between system releases. However, software evolves over time, and during development not only bugs are fixed, but also features are added, and code is refactored. In such cases, changes must be consistently applied in source code to avoid maintenance problems. We propose to extract system specific rules by mining systematic changes over source code history, i.e., not just from bug-fixes or system releases, to ensure that changes are consistently applied over source code. We focus on structural changes done to support API modification or evolution with the goal of providing better rules to developers. Also, rules are mined from predefined rule patterns that ensure their quality. In order to assess the precision of such specific rules to detect real violations, we compare them with generic rules provided by tools to detect coding standard violations on four real world systems covering two programming languages. The results show that specific rules are more precise in identifying real violations in source code than generic ones, and thus can complement them. [25]

6.4. Reconciling Dynamic Languages and Isolation

Virtual Smalltalk Images: Model and Applications. Reflective architectures are a powerful solution for code browsing, debugging or in-language process handling. However, these reflective architectures show some limitations in edge cases of self-modification and self-monitoring. Modifying the modifier process or monitoring the monitor process in a reflective system alters the system itself, leading to the impossibility to perform some of those tasks properly. We analyze the problems of reflective architectures in the context of image based object-oriented languages and solve them by providing a first-class representation of an image: a virtualized image. We present Oz, our virtual image solution. In Oz, a virtual image is represented by an object space. Through an object space, an image can manipulate the internal structure and control the execution of other images. An Oz object space allows one to introspect and modify execution information such as processes, contexts, existing classes and objects. We show how Oz solves the edge cases of reflective architectures by adding a third participant, and thus, removing the self modification and self-observation constraints. [30]

Bootstrapping Reflective Systems: The Case of Pharo. Bootstrapping is a technique commonly known by its usage in language definition by the introduction of a compiler written in the same language it compiles. This process is important to understand and modify the definition of a given language using the same language, taking benefit of the abstractions and expression power it provides. A bootstrap, then, supports the evolution of a language. However, the infrastructure of reflective systems like Smalltalk includes, in addition to a compiler, an environment with several self-references. A reflective system bootstrap should consider all its infrastructural components. We propose a definition of bootstrap for object-oriented reflective systems, we describe the architecture and components it should contain and we analyze the challenges it has to overcome. Finally, we present a reference bootstrap process for a reflective system and Hazelnut, its implementation for bootstrapping the Pharo Smalltalk-inspired system. [15]

Object Graph Isolation with Proxies More and more software systems are now made of multiple collaborating third-party components. Enabling fine-grained control over the communication between components becomes a major requirement. While software isolation has been studied for a long time in operating systems (OS), most programming languages lack support for isolation. In this context we explore the notion of proxy. A proxy is a surrogate for another object that controls access to this object. We are particularly interested in generic proxy implementations based on language-level reflection. We present an analysis that shows how these reflective proxies can propagate a security policy thanks to the transitive wrapping mechanism. We present a prototype implementation that supports transitive wrapping and allows a fine-grained control over an isolated object graph. [33]

6.5. Dynamic Languages: Compilers

Towards a flexible Pharo Compiler The Pharo Smalltalk-inspired language and environment started its development with a codebase that can be traced back to the original Smalltalk-80 release from 1983. Over the last years, Pharo has been used as the basis of many research projects. Often these experiments needed

changes related to the compiler infrastructure. However, they did not use the existing compiler and instead implemented their own experimental solutions. This shows that despite being an impressive achievement considering its age of over 35 years, the compiler infrastructure needs to be improved. We identify three problems: (i) The architecture is not reusable, (ii) compiler can not be parametrized and (iii) the mapping between source code and bytecode is overly complex. Solving these problems will not only help researchers to develop new language features, but also the enhanced power of the infrastructure allows many tools and frameworks to be built that are important even for day-to-day development, such as debuggers and code transformation tools. [20]

Gradual Typing for Smalltalk Being able to combine static and dynamic typing within the same language has clear benefits in order to support the evolution of prototypes or scripts into mature robust programs. While being an emblematic dynamic object-oriented language, Smalltalk is lagging behind in this regard. We report on the design, implementation and application of Gradualtalk, a gradually-typed Smalltalk meant to enable incremental typing of existing programs. The main design goal of the type system is to support the features of the Smalltalk language, like metaclasses and blocks, live programming, and to accomodate the programming idioms used in practice. We studied a number of existing projects in order to determine the features to include in the type system. As a result, Gradualtalk is a practical approach to gradual types in Smalltalk, with a novel blend of type system features that accomodate most programming idioms. [13]

7. Bilateral Contracts and Grants with Industry

7.1. Resilience FUI

Participants: Nicolas Petton [Correspondant], Stéphane Ducasse, Damien Cassou.

Contracting parties: Nexedi, Morphom Alcatel-Lucent Bell Labs, Astrium Geo Information, Wallix, XWiki, Alixen, Alterway, Institut Télécom, Université Paris 13, CEA LIST.

Resilience's goal is to protect private data on the cloud, to reduce spying and data loss in case of natural problems. Resilience proposes to develop a decentralized cloud architecture: SafeOS. Safe OS is based on replication of servers. In addition a safe solution for documents should be developed. Sandboxing for Javascript applications should be explored.

There is a plethora of research articles describing the deep semantics of JavaScript. Nevertheless, such articles are often difficult to grasp for readers not familiar with formal semantics. In our first report, we propose a digest of the semantics of JavaScript centered around security concerns. This report proposes an overview of the JavaScript language and the misleading semantic points in its design.

7.2. SafePython FUI

Participants: Damien Cassou [Correspondant], Jean-Baptiste Arnaud, Stephane 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.

7.3. Generali France

Participants: Nicolas Anquetil [Correspondant], Stéphane Ducasse, Guillaume Larcheveque, Muhammad Bhatti, Camille Teruel.

Contracting parties: Synectique, Generali Assurances http://www.generali.be.

RMoD is looking into providing a software solution to Generali France for its software maintenance. The goal is to support decision making by providing quality metrics and softare dependance information. The partner needs tools for parsing their legacy code (in a specific, not well-known language) and help in assessing quality and identifying dead code or code duplication. This should serve as an essential element of decision support in the continuing evolution of an important software system of the partner.

7.4. Pharo Consortium

We launched the Pharo Consortium. It has 13 members, 6 academic partners and 3 sponsoring companies. Inria supports the consortium with one full time engineer starting in 2011. More at http://consortium.pharo.org.

8. Partnerships and Cooperations

8.1. Regional Initiatives

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

8.2. National Initiatives

8.2.1. ANR

8.2.1.1. Cutter

Participants: Stéphane Ducasse [Correspondant], Nicolas Anquetil, Damien Pollet, Muhammad Bhatti, Andre Calvante Hora.

This parternship is done with the following members from the LIRMM-D'OC-APR: Marianne Huchard, Roland Ducournau, Jean-Claude König, Rodokphe Giroudeau, Abdelhak-Djamel Seriai, and Rémi Watrigant.

CUTTER is a Basic Research project that addresses the problems of object-oriented system remodularization by developing, combining, and evaluating new techniques for analyzing and modularizing code. In particular, it will: (i) use concurrently and collaboratively four package decomposition techniques; and (ii) take into account different levels of abstractions (packages, classes).

8.3. European Initiatives

8.3.1. IAP MoVES

Participant: Stéphane Ducasse [correspondant].

The Belgium IAP (Interuniversity Attraction Poles) MoVES (Fundamental Issues in Software Engineering: Modeling, Verification and Evolution of Software) is a project whose partners are the Belgium universities (VUB, KUL, UA, UCB, ULB, FUNDP, ULg, UMH) and three European institutes (Inria, IC and TUD) respectively from France, Great Britain and Netherlands. This consortium combines the leading Belgian research teams and their neighbors in software engineering, with recognized scientific excellence in MDE, software evolution, formal modeling and verification, and AOSD. The project focusses on the development, integration and extension of state-of-the-art languages, formalisms and techniques for modeling and verifying dependable software systems and supporting the evolution of Software-intensive systems. The project has started in January 2007 and is scheduled for a 60-months period. Read more at http://moves.vub.ac.be.

8.3.2. 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.

8.3.3. 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 Marie Curie's 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

Visits in the context of MEALS

- Guido Chari visited RMoD from 29/11/2013 to 22/12/2013
- Diego Garbervetsky visited RMoD from 16/12/2013 to 17/12/2013
- Camillo Bruni visited UBA (Buenos Aires, Argentina): 2012-09-01 - 2012-09-30

8.4. International Initiatives

8.4.1. Inria Associate Teams

8.4.1.1. PLOMO

Title: Customizable Tools and Infrastructure for Software Development and Maintenance

Inria principal investigator: Stéphane Ducasse

International Partner (Institution - Laboratory - Researcher):

University of Chile (Chile) - PLEIAD

Duration: 2011–2013

See also: http://pleiad.dcc.uchile.cl/research/plomo

Project Description

Software maintenance is the process of maintaining a software system by removing bugs, fixing performance issues and adapting it to keep it useful and competitive in an ever-changing environment [50]. Performing effective software maintenance and development is best achieved with effective tool support, provided by a variety of tools, each one presenting a specific kind of information supporting the task at hand [52]. The goal of PLOMO is to develop new meta tools to improve and bring synergy in the existing infrastructure of Pharo (for software development) and the Moose software analysis platform (for software maintenance).

PLOMO will (1) enhance the Opal open compiler infrastructure to support plugin definition, (2) offer an infrastructure for change and event tracking as well as model to compose and manipulate them, (3) work on a layered library of algorithms for the Mondrian visualization engine of Moose, (4) work on new ways of profiling applications. All the efforts will be performed on Pharo and Moose, two platforms heavily used by the RMoD and PLEIAD team.

The artifacts produced by PLOMO will strongly reinforce the Pharo programming language and the Moose software analysis platform. The development and progress of Pharo is structured by RMoD, which has successfully created a strong and dynamic community. Moose is being used to realize consulting activities and it is used as a research platform in about 10 Universities, worldwide. We expect PLOMO to have a strong impact in both the software products and the communities structured around them.

2013 was the third and final year of PLOMO. Due to the success of PLOMO, we have requested a prolongation for another three years (PLOMO2). The *PLOMO Associate Team Final Report* is available online [37].

In the following, we present the results from 2013:

Research Visits From RMoD to PLEIAD

- Stéphane Ducasse from November 4 until November 15, 2013.

From PLEIAD to RMoD

- Johan Fabry on 15th of July, 18th and 19th of September 2013
- Alexandre Bergel from December 12 until December 29, 2013
- Alejandro Infante from September 13 until September 21, 2013
- Ronie Salgado in January 2014

Recent Results

In the third year of execution of Plomo, work has focused on:

- GradualTalk Paper accepted at Science of Computer Programming.
- Performance Evolution Blueprint paper at VISSOFT.
- Work on the DIE domain-specific language and the definition of IDE plugins using it was submitted to a journal and is in a second round of revisions.
- Organization of a coding sprint at Santiago in January 2013 (12 participants)
- Participated to three Moose releases (4.7-4.9) (http://www.moosetechnology.org)
- Integrated the Opal Compiler in the Pharo3 development branch.

Future of the Partnership We really hope that the team will be prolongated for a second three year period. The synergy between the two teams is working really well - in terms of exchanges, results and future collaborations.

For more information, we refer to the report *PLOMO Associate Team Final Report* [37].

8.4.2. Inria International Partners

8.4.2.1. Uqbar - Argentina

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

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).

8.4.2.2. Informal International Partners

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.

8.4.3. Participation In other International Programs

8.4.3.1. Project Pequi – Inria/CNPq Brésil

The Pequi project is a collaboration between Professor Marco T. Valente's team at the Federal University of Minas Gerais in Brazil and the RMoD team. It focuses in producing Metrics, Techniques, and Tools for Software Remodularization.

It is recognized that software systems must be continuously maintained and evolved to remain useful. However, ongoing maintenance over the years contributes to degrade the quality of a system. Thus reengineering activities, including remodularization activities, are necessary to restore or enhance the maintainability of the systems. To help in the remodularization of software systems, the project will be structured in two main research lines in which both teams have experience and participation: (i) Evaluation and Characterization of Metrics for Software Remodularization; and (ii) Tools and Techniques for Removal of Architectural Violations.

The project started in July 2011 with a visit of Dr. Nicolas Anquetil to the brazilian team. The project lasted 24 months and ended June 2013.

8.5. International Research Visitors

8.5.1. Visits of International Scientists

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

- Johan Fabry on 15th of July, 18th and 19th of September 2013
- Alexandre Bergel from December 12 until December 29, 2013
- Alejandro Infante from September 13 until September 21, 2013
- Ronie Salgado in January 2014

In the context of the Pequi project associated Team with the Federal University of Minas Gerais:

- Marco Tulio Valente from 21/01/2013 to 25/01/2013
- Marco Tulio Valente from 22/07/2013 to 26/07/2013

In the context of MEALS:

- Guido Chari visited RMoD from 29/11/2014 to 22/12/2013.
- Diego Garbervetsky visited RMoD 16 and 17 December.

Other visitors:

- Hani Abdeen, Research Associate at Computer Science Department - Qatar University (January 2013)
- Michele Lanza, Professor at the University of Lugano (2nd of May, 2013)
- Hayatou Oumarou, Assistant Departement d'Informatique ENS Maroua Cameroun (March 2013 for 1 Month
- David Chisnall, Research Associate at University of Cambridge, (4-5 February)
- Tommaso Dal Sasso, University of Lugano, (16-22 December)
- Yuriy Tymchuk, University of Lugano, (16-22 December)
- Roberto Minelli, University of Lugano, (16-22 December)
- Andrei Vasile, University of Bern, Switzerland (18-21 December)
- Jan Kurs, University of Bern, Switzerland, (18-21 December)

8.5.1.1. Internships

Pablo Herrero, University of Buenos Aires (Argentina): *Compressed ASTs for Pharo*, from Oct 2013.

Sebastian Tleye, University of Buenos Aires (Argentina): *A new Trait Implementation*, from Mar until Aug 2013.

Gustavo Jansen De Souza Santos, Federal University of Minas Gerais (Brasil): *Integration of Semantic Clustering in Moose*, September until November 2013.

Gisela Decuzzi, Universidad Tecnolo´gica Nacional FRBA (Argentina): *AST Navigation for Pharo*, from March to May 2013.

Yuriy Tymchuk, Ivan Franko National University of Lviv (Ukraine): *Extending FAMIX metamodel to generate ASTs for Java and Smalltalk applications*, from January to April 2013.

Erwan Douaille, University of Lille 1: *Automatic validation of contributions from Pharo community*, From April to Jun 2013

8.5.2. Visits to International Teams

- Stéphane Ducasse from November 4 until November 15, 2013. (PLOMO).

- Stéphane Ducasse from November to University of Buenos Aires (Argentina) (MEALS).
- Stéphane Ducasse and Igor Stasenko visited the University of Lviv, Ukraine.
- Camillo Bruni visited UBA (Buenos Aires, Argentina): September (1 Month)
- Marcus Denker visited *Universitat Politecnica de Catalunya*, Barcelona, Spain, 1 week in October 2013.
- Stéphane Ducasse visted the University of Prag for one week in December, 2013

9. Dissemination

9.1. Scientific Animation

9.1.1. Organization

- Guillermo Polito helped organizing the *Uqbar Workshop* (November 2013) in Argentina.
- Damien Cassou: DYLA 2013, *7th Workshop on Dynamic Languages and Applications*. Colocated with ECMFA, ECOOP and ECSA, 1–5 July, Montpellier, France.
- Stéphane Ducasse and Marcus Denker: Part of the organizaton team of ESUG 2013, *21th International Smalltalk Conference*, Annecy, France 9-13 September 2013.

9.1.2. PC Memberships and Reviewing

- Anne Etien
  - ECMFA 2013, *European Conference on Modelling Foundations and Applications*.
  - IWST 2013, *International Workshop on Smalltalk Technologies*.
  - ME 2013, *Models and Evolution*
  - SLE 2013, *International Conference on Software Language Engineering*.
- Marcus Denker
  - DLS 2013, *Dynamic Languages Symposium at SPLASH*.
  - Varicomp 2013, *International Workshop on Variability and Composition*.
  - ICSM 2013 Tool Demo Track, *International Conference on Software Maintance*.
  - SCAM 2013 Tool Paper Track,*IEEE International Working Conference on Source Code Analysis and Manipulation*.
  - SLE 2013, *International Conference on Software Language Engineering*. External Reviewer.
- Damien Cassou
  - aec-esec FSE 2013, *ACM SIGSOFT Symposium on the Foundations of Software Engineering*.
  - ASE 2013, *28th IEEE/ACM International Conference on Automated Software Engineering*.
  - DYLA 2013, *7th Workshop on Dynamic Languages and Applications*.
  - WASDETT 2013, *4th International Workshop on Academic Software Development Tools and Techniques*.
- Nicolas Anquetil:
  - ICSM 2013, *International Conference on Software Maintance*.
  - ECOOP/ECMFA/ECSA 2013 Doctoral Symposium.
  - Reviewer EMSE (Empirical Software Engineering journal).

– Reviewer SCICO (Science of Computer Programming journal).

9.1.3. Memberships

- Marcus Denker is a member of *DFGWT - Deutsch-Französische Gesellschaft für Wissenschaft und Technologie e.V* (Associaton Franco-Allemande pour la Science et la Technologie) (from August 2013).
- Nicolas Anquetil and Anne Etien are member of IEEE. (from July 2013).

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Master: C. Demarey, Architectures Logicielles, 30h (M1), GIS4, Polytech'Lille, France

Misc: Stéphane Ducasse , Ecoles des jeunes chercheurs Inria (2013), 6h, Inria, France

Licence: Stéphane Ducasse, Cours IAE (2013), 14h, Annecy, France

Master: Stéphane Ducasse, Cours Ecole des Mines de Douai (2013), 12h, France

Master: Stéphane Ducasse, Cours USTL L3 (2013), 4.5h, Université Lille, France

Master: Stéphane Ducasse, Cours University of Quilmes, 2h, Argentina

Master: Stéphane Ducasse, Cours University of Lviv, 12h, Ukraine

Master: Stéphane Ducasse, Cours University of Buenos Aires, 3h, Argentina

Master: Stéphane Ducasse, Cours FIT Prague (dec 2013), 8h, Czech Republic

Licence: Nicolas Anquetil, Software engineering: Testing techniques, 40h, (L2), IUT-A Lille, France

Licence: Nicolas Anquetil, Interface programming in Java/Swing, 48h, (L2), IUT-A Lille, France

Licence: Anne Etien, Java, Course: 30h (L3), Polytech Lille, France

Master: Anne Etien, Object Information System, TP: 8h (M1), Polytech Lille, France

Master: Damien Cassou, Algo, 24h, Université Lille 1, France

Master: Damien Cassou, COO, 18h, Université Lille 1, France

Master: Damien Cassou, [OPL] Test, maintenance & évolution, 12h, Université Lille 1, France

Master: Damien Cassou, Qualité logicielle, 30h, Universite´ Lille 1, France

Master: Damien Cassou, CAR, 24h, Université Lille 1, France

Master: Damien Cassou, COA, 24h, Université Lille 1, France

License: Damien Pollet, UV info, 100h, Introduction to OO programming in Java, Telecom Lille, France

Master: Damien Pollet, SER, 25h, Distributed Algorithms, Telecom Lille, France

Master: Damien Pollet, ILOG & TSIO, 35h, Programming & Software engineering, Telecom Lille, France

Master: Damien Pollet, Supervision of industry internships, Telecom Lille, France

Master: Guillermo Polito, Cours University of Quilmes, 2h, University of Quilmes, Argentina

9.2.2. Supervision

PhD: Jean Baptiste Arnaud, Towards First Class References as a Security Infrastructure in Dynamic Languages, 18/02/2013, Marcus Denker, Stéphane Ducasse

PhD: Nick Papoylias, Languages and Development Environments for Mobile Autonomous Robots, 19/12/2013, Marcus Denker, Stéphane Ducasse

PhD in progress: Camillo Bruni, Entering the 4th Quadrant, 01/03/2011, Stéphane Ducasse, Marcus Denker

PhD in progress: Andre Hora, Improving Static Analysis with Domain-Specific Rules, 01/12/2011, Nicolas Anquetil, Stéphane Ducasse

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

PhD in progress: Martin Dias, Supporting Merging, 03/12/2012, Damien Cassou, Stéphane Ducasse

PhD in progress: Guillermo Polito, Isolation and Reflection in Dynamic Object Oriented Languages, 01/04/2012, Noury Bouraqadi, Luc Fabrese, Marcus Denker, Stéphane Ducasse

9.2.3. Juries

Stéphane Ducasse was in the examination committee of the following PhD theses:

- F. Olivero, Lugano, 02/2013 (reviewer)
- Q. Sabah, Grenoble 12/2013 (examinator)

Anne Etien was in the examination committee of the following PhD theses:

- D. Blouin, Lorient, 10/12/2013

9.3. Popularization

- Marcus Denker gave a presentation about Pharo at the open source conference FOSDEM 2013.
- *Web with Pharo* Conference was held 6 June 2013 @ Euratechnologies, Lille.
- Camille Teruel and Damien Cassou gave a Pharo tutorial at ECOOP 2013 Montpellier. Slides got >16000 hits after a mention on ycombinator news (http://www.lirmm.fr/ecoop13/?id=158#tutorial13)
- *RIC Day* was held 2 October 2013 @ University of Lille 1, Lille, Anne Etien gave a presentation about Moose and software maintenance.
- Tutorials at ESUG 2013, Annecy, France in September.
- Esteban Lorenzano and Guillermo Polito gave multiple Presentations at Smalltalks 2013 (Argentina):
- Guillermo Polito gave a lecture about *Pharo Ninja Tricks* at Universidad de Quilmes / Argentina in November (a hands-on showing how to use the IDE and explore the system, open classes, extension methods...).
- Marcus Denker gave a lecture and a public talk at *Universitat Politecnica de Catalunya*, Barcelona, Spain.
- Software Maintenance Courses (Nicolas Anquetil):
  - IJDs Inria Lille Nord Europe, with Guillaume Larcheveque, 1 day
  - Thales, 1 day
- Stéphane Ducasse gave multiple lectures and talks at University of Prag in December 2013.
- Organization of the MooseDay in Lille on the 19th December with around 25 persons from all around the world. (http://www.lifl.fr/~etien/MooseDay.html).
- Multiple public Pharo Sprints in Lille, Santiago/Chile, Buenos Aires/Argentina and other places.

10. 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° 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° 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° 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° 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, pp. 21–., 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. [*DOI : 10.1145/1085313.1085331*]

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] J.-B. ARNAUD. *Vers des références de première classe comme infrastructure de sécurité dans les langages dynamiquement typés*, Université des Sciences et Technologie de Lille - Lille I, February 2013., http://hal.inria.fr/tel-00808419

[12] N. PAPOULIAS. *Le Débogage à Distance et la Réflexion dans les Dispositifs à Ressources Limitées*, Université des Sciences et Technologie de Lille - Lille I, December 2013., http://hal.inria.fr/tel-00932796

Articles in International Peer-Reviewed Journals

[13] E. ALLENDE, O. CALLAU, J. FABRY, É. TANTER, M. DENKER. *Gradual Typing for Smalltalk*, in "Science of Computer Programming", August 2013. [*DOI :* 10.1016/J.SCICO.2013.06.006], http://hal.inria.fr/hal-00862815

[14] M. MARTINEZ PECK, N. BOURAQADI, M. DENKER, S. DUCASSE, L. FABRESSE. *Marea: An Efficient Application-Level Object Graph Swapper*, in "Journal of Object Technology", January 2013, vol. 12, n^{o} 1, pp. 2:1-30. [*DOI :* 10.5381/JOT.2013.12.1.A2], http://hal.inria.fr/hal-00781129

[15] G. POLITO, S. DUCASSE, L. FABRESSE, N. BOURAQADI, B. VAN RYSEGHEM. *Bootstrapping Reflective Systems: The Case of Pharo*, in "Science of Computer Programming", January 2014., http://hal.inria.fr/hal-00903724

[16] V. UQUILLAS-GOMEZ, S. DUCASSE, T. D'HONDT. *Visually Characterizing Source Code Changes*, in "Science of Computer Programming", September 2013., http://hal.inria.fr/hal-00862049

[17] B. VAN RYSEGHEM, S. DUCASSE, J. FABRY. *Seamless Composition and Reuse of Customizable User Interfaces with Spec*, in "Science of Computer Programming", June 2014., http://hal.inria.fr/hal-00915350

Articles in National Peer-Reviewed Journals

[18] A. AUTHOSSERRE-CAVARERO, F. BERTRAND, M. FORNARINO, P. COLLET, H. DUBOIS, S. DUCASSE, S. DUPUY-CHESSA, C. FARON-ZUCKER, C. FAUCHER, J.-Y. LAFAYE, P. LAHIRE, O. LE GOAER, J. MONTAGNAT, A.-M. PINNA-DÉRY. *Ingénierie dirigée par les modèles : quels supports à l'interopérabilité des systèmes d'information ?*, in "REVUE INGENIERIE DES SYSTEMES D'INFORMATION", April 2013., http://hal.inria.fr/hal-00813675

International Conferences with Proceedings

[19] H. ABDEEN, H. SAHRAOUI, O. SHATA, N. ANQUETIL, S. DUCASSE. *Towards Automatically Improving Package Structure While Respecting Original Design Decisions*, in "Working Conference on Reverse Engineering", Glasgow, United Kingdom, September 2013., http://hal.inria.fr/hal-00862063

[20] C. BERA, M. DENKER. *Towards a flexible Pharo Compiler*, in "IWST", Annecy, France, L. LAGADEC, A. PLANTEC (editors), ESUG, September 2013., http://hal.inria.fr/hal-00862411

[21] C. BRUNI, S. DUCASSE, I. STASENKO, L. FABRESSE. *Language-side Foreign Function Interfaces with NativeBoost*, in "International Workshop on Smalltalk Technologies", Annecy, France, September 2013., http://hal.inria.fr/hal-00840781

[22] C. COUTO, P. PIRES, M. T. VALENTE, R. BIGONHA, A. HORA, N. ANQUETIL. *BugMaps-Granger: A Tool for Causality Analysis between Source Code Metrics and Bugs*, in "Brazilian Conference on Software: Theory and Practice (CBSoft'13)", Brasilia, Brazil, September 2013., http://hal.inria.fr/hal-00854883

[23] M. DEHOUCK, M. U. BHATTI, A. BERGEL, S. DUCASSE. *Pragmatic Visualizations for Roassal: a Florilegium*, in "International Workshop on Smalltalk Technologies", Annecy, France, September 2013., http://hal.inria.fr/hal-00862065

[24] M. DIAS, D. CASSOU, S. DUCASSE. *Representing Code History with Development Environment Events*, in "IWST-2013 - 5th International Workshop on Smalltalk Technologies", Annecy, France, September 2013., http://hal.inria.fr/hal-00862626

[25] A. HORA, N. ANQUETIL, S. DUCASSE, M. T. VALENTE. *Mining System Specific Rules from Change Patterns*, in "Working Conference on Reverse Engineering (WCRE'13)", Koblenz, Germany, October 2013., http://hal.inria.fr/hal-00854861

[26] C. MAFFORT, M. T. VALENTE, N. ANQUETIL, A. HORA, M. BIGONHA. *Heuristics for Discovering Architectural Violations*, in "Working Conference on Reverse Engineering (WCRE'13)", Koblenz, Germany, October 2013., http://hal.inria.fr/hal-00854871

[27] C. MAFFORT, M. T. VALENTE, M. BIGONHA, A. HORA, N. ANQUETIL, J. MENEZES. *Mining Architectural Patterns Using Association Rules*, in "International Conference on Software Engineering and Knowledge Engineering (SEKE'13)", Boston, United States, June 2013., http://hal.inria.fr/hal-00854851

[28] D. MENDEZ ACUNA, R. CASALLAS, A. ETIEN. *On the customization of model management systems for file-centric IDEs*, in "The 13th Workshop on Domain-Specific Modeling", United States Minor Outlying Islands, October 2013, pp. pp 57-62., http://hal.inria.fr/hal-00913246

[29] P. PATEL, A. PATHAK, D. CASSOU, V. ISSARNY. *Enabling High-Level Application Development in the Internet of Things*, in "4th International Conference on Sensor Systems and Software", Lucca, Italy, April 2013., http://hal.inria.fr/hal-00809438

[30] G. POLITO, S. DUCASSE, L. FABRESSE, N. BOURAQADI. *Virtual Smalltalk Images: Model and Applications*, in "21th International Smalltalk Conference - 2013", Annecy, France, July 2013, pp. 11-26., http://hal.inria.fr/hal-00924932

[31] J. P. SANDOVAL ALCOCER, A. BERGEL, S. DUCASSE, M. DENKER. *Performance Evolution Blueprint: Understanding the Impact of Software Evolution on Performance*, in "VISSOFT - 1st IEEE Working Conference on Software Visualization", Eindhoven, Netherlands, A. C. TELEA (editor), IEEE, September 2013, pp. 1-9. [*DOI :* 10.1109/VISSOFT.2013.6650523], http://hal.inria.fr/hal-00849004

[32] G. SANTOS, M. TULIO VALENTE, N. ANQUETIL. *Remodularization Analysis Using Semantic Clustering*, in "1st CSMR-WCRE Software Evolution Week", Antwerp, Belgium, 2014., http://hal.inria.fr/hal-00904409

[33] C. TERUEL, D. CASSOU, S. DUCASSE. *Object Graph Isolation with Proxies*, in "DYLA - 7th Workshop on Dynamic Languages and Applications, Collocated with 26th European Conference on Object-Oriented Programming - 2013", Montpellier, France, 2013., http://hal.inria.fr/hal-00834320

Scientific Books (or Scientific Book chapters)

[34] A. BERGEL, D. CASSOU, S. DUCASSE, J. LAVAL. *Deep into Pharo*, Square Bracket Associates, September 2013, 420., http://hal.inria.fr/hal-00858725

[35] C. DENKER, N. HARTL, M. DENKER. *Kapitel 1: Apps*, in "Mobile Apps - Rechtsfragen und rechtliche Rahmenbedingungen", C. SOLMECKE, J. TAEGER, T. FELDMANN (editors), De Gruyter, 2013, pp. 1-8., http://hal.inria.fr/hal-00865117

[36] N. HARTL, C. DENKER, M. DENKER. *Kapitel 2: Technische Aspekte*, in "Mobile Apps - Rechtsfragen und rechtliche Rahmenbedingungen", C. SOLMECKE, J. TAEGER, T. FELDMANN (editors), De Gruyter, 2013, pp. 9-24., http://hal.inria.fr/hal-00865116

Research Reports

[37] A. BERGEL, S. DUCASSE, M. DENKER, J. FABRY. *PLOMO Associate Team Final Report*, October 2013., http://hal.inria.fr/hal-00871114

[38] D. CASSOU, S. DUCASSE, N. PETTON. *SafeJS: Hermetic Sandboxing for JavaScript*, September 2013, 7., http://hal.inria.fr/hal-00862099

Other Publications

[39] J.-B. ARNAUD, S. DUCASSE, M. DENKER. *Handles: Behavior-Propagating First Class References For Dynamically-Typed Languages*, November 2013, Preprint, Accepted with minor revisions., http://hal.inria.fr/hal-00881865

[40] G. CHARI, D. GARBERVETSKY, C. BRUNI, M. DENKER, S. DUCASSE. *Waterfall: Primitives Generation on the Fly*, 2013., http://hal.inria.fr/hal-00871353

[41] M. MARTINEZ PECK, N. BOURAQADI, S. DUCASSE, L. FABRESSE, M. DENKER. *Ghost: A Uniform and General-Purpose Proxy Implementation*, October 2013, Preprint., http://hal.inria.fr/hal-00877757

References in notes

[42] 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

[43] 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

[44] 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

[45] 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^{o} 2-3, pp. 83–108., http://dx.doi.org/10.1016/j.cl.2007.05.003

[46] 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

[47] 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

[48] 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.

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

[50] E. CHIKOFSKY, J. CROSS II. *Reverse Engineering and Design Recovery: A Taxonomy*, in "IEEE Software", January 1990, vol. 7, n^{o} 1, pp. 13–17., http://dx.doi.org/10.1109/52.43044

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

[52] S. DEMEYER, S. DUCASSE, O. NIERSTRASZ. *Object-Oriented Reengineering Patterns*, Morgan Kaufmann, 2002., http://www.iam.unibe.ch/~scg/OORP

[53] 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.

[54] 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

[55] 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, Milano, 2005, pp. 55–71., http://scg.unibe.ch/archive/papers/Duca05aMooseBookChapter.pdf

[56] 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

[57] 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

[58] 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.

[59] 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

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

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

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

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

[64] 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

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

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

[67] R. KOSCHKE. *Atomic Architectural Component Recovery for Program Understanding and Evolution*, Universität Stuttgart, 2000., http://www.informatik.uni-stuttgart.de/ifi/ps/bauhaus/papers/koschke.thesis.2000.html

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

[69] 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

[70] 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.

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

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

[73] 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.

[74] 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

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

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

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

[78] 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.

[79] 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.

[80] 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

[81] 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

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

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

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

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

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