Project-Team RMoD 2014 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 2014

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, HdR]
Damien Cassou [Univ. Lille I, Associate Professor]
Anne Etien [Univ. Lille I, Associate Professor]
Damien Pollet [Univ. Lille I, Associate Professor]
Jean-Christophe Bach [Univ. Lille I, ATER, from Sep 2014]

Engineers

Olivier Auverlot [Univ. Lille I, Engineer 20%]
Christophe Demarey [Inria, Engineer 60%]
Muhammad Bhatti [Inria, until Jan 2014]
Guillaume Larcheveque [Inria, until Jul 2014, granted by SYNECTIQUE]
Esteban Lorenzano [Inria]
Nicolas Petton [Inria, until Jul 2014, FUI Resilience]
Igor Stasenko [Inria, until Nov 2014]

PhD Students

Clément Bera [Inria, started Oct 2014]
Camillo Bruni [Inria, until Apr 2014]
Andre Cavalcante Hora [Inria, until Nov 2014]
Camille Teruel [Inria]
Victor Martin Dias [Inria]
Max Mattone [Ecole des Mines de Douai, co-supervision, from Nov 2014]
Guillermo Polito [Ecole des Mines de Douai, co-supervision]
Gustavo Jansen de Souza Santos [Brazilian Government, from Jan 2014]
Marco Naddeo [University of Turin, co-supervision, from Jan 2014]
Vincent Blondeau [CIFRE with WordLine, from Oct 2014]
Brice Morin [CIFRE with Thales, start Jan 2015]

Post-Doctoral Fellows

Jean-Baptiste Arnaud [Inria, granted by FUI SafePython, until Dec 2014]
Stefan Marr [Inria]
Ricardo Terra Nunes Bueno Villela [Visiting Postdoc, from Dec 2014]

Visiting Scientists

Abdelghani Alidra [Associate Professor, from Jun 2014 until Jul 2014]
Rafael Serapilha Durelli [Visiting PhD Student, University of Sao Paulo, Brazil, until Mar 2014]
Miguel Campusano Araya [Visiting PhD student, from Jan 2015 to Jul 2015]
Luciana Silva [Visiting PhD student, from Oct 2014 to Dec 2014]

Administrative Assistant

Julie Jonas [Inria]

Others

Benjamin Van Ryseghem [Student, Univ. Lille I, until Jul 2014]
Baptiste Quide [Inria, Intern, from May 2014 until Aug 2014]
Kevin Lanvin [Inria, Intern, from Jan 2014 until Apr 2014]
Hayatou Oumarou [Inria, Intern, until Oct 2014]
Clara Allende [Inria, Intern, from May 2014 until Oct 2014]
Pablo Herrero [Inria, Intern, until Feb 2014]
Lucas Ariel Godoy [Inria, Intern, from May 2014 until Oct 2014]
Markiyan Rizun [Intern, from Jul 2014 to Aug 2014]
Leo Perard [Inria, Intern, from Mar 2014 until Aug 2014]

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 [65]. 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 [62]. 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 [71]. 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?

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 [56], [55]. 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) [84] show that such an analysis can be used to identify modules. However the presented examples are too small and not representative of real code.

[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.

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 [45]. 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 [72]. Until now, few works have attempted to identify layers in practice: Mudpie [86] is a first cut at identifying cycles between packages as well as package groups potentially representing layers. DSM (dependency structure matrix) [85], [80] 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 [68]. 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 [87], [59].

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 [82], [57] and classboxes [46] 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 [82], [57]. 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 [54], [76], [47], [58] and several type systems were defined [60], [83], [77], [70].

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 [49]. 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 [57], stateful [48], and freezable [58]) 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 [75]. 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 [53] then from Smalltalk [63].

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 [67]). 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 [50], as well as controlling the access to reflective features [51], [52] are important challenges. We plan to:

- Study the isolation abstractions available in erights (http://www.erights.org) [74], [73], and Java's class loader strategies [69], [64].
- Categorize the different reflective features of languages such as CLOS [66], Python and Smalltalk [78] and identify suitable isolation mechanisms and infrastructure [61].
- Assess different isolation models (access rights, capabilities [79]...) 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 [50],
  - the use of encapsulation policies as a basis to restrict the interfaces of the controlled objects [81],
  - 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...) [78].

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) 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. New Software and Platforms

5.1. Moose5.0

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. The current focus is Moose 5.0, in late beta testing as of December 2014. A major development of 2014 is that tools and frameworks built for Moose are being integrated into Pharo4 as the default development tools.
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. Pharo3.0

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 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.

The first stable version, Pharo 1.0, was released in 2010. We are now releasing one new version of Pharo every year, with Pharo3 released in spring 2014. Pharo4 has seen already over 400 incremental updates and is scheduled for early 2015. It should be noted that Pharo, even though already used outside of research, still continues to improve radically.

In November 2012 RMoD launched the Pharo Consortium (http://consortium.pharo.org/) and the Pharo Association (http://association.pharo.org). The consortium has now 14 industrial members, 3 sponsors and 10 academic partners.

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 in place.
- **(EM4)** Evolution and Maintenance: 4 – Active user group, consortium and association has 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. Pillar 0.17

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. http://www.smalltalkhub.com/#!/~Pier/Pillar

6. New Results

6.1. Highlights of the Year

- Pharo 3.0 has been released in April 2014.
- Moose 5.0 has been released in December 2014.
- The book Deep into Pharo has been released publicly http://www.deepintopharo.com.
- RMOD entered in a sponsoring agreement with LAM Research, Inc.

6.2. Tools for understanding applications

Remodularization Analysis Using Semantic Clustering. We report an experience on using and adapting Semantic Clustering to evaluate software remodularizations. Semantic Clustering is an approach that relies on information retrieval and clustering techniques to extract sets of similar classes in a system, according to their vocabularies. We adapted Semantic Clustering to support remodularization analysis. We evaluate our adaptation using six real-world remodularizations of four software systems. We report that Semantic Clustering and conceptual metrics can be used to express and explain the intention of the architects when performing common modularization operators, such as module decomposition. [37]

Towards a new package dependency model. Smalltalk originally did not have a package manager. Each Smalltalk implementation defined its own with more or less functionalities. Since 2010, Monticello/Metacello[Hen09] one package manager is available for open-source Smalltalks. It allows one to load source code packages with their dependencies. This package manager does not have all features we can find in well-known package managers like those used for the Linux operating system. We identify the missing features and propose a solution to reach a full-featured package manager. A part of this solution is to represent packages and dependencies as first-class objects, leading to the definition of a new dependency model. [32]

A Domain Specific Aspect Language for IDE Events. Integrated development environments (IDEs) have become the primary way to develop software. Besides just using the built-in features, it becomes more and more important to be able to extend the IDE with new features and extensions. Plugin architectures exist, but they show weaknesses related to unanticipated extensions and event handling. We argue that a more general solution for extending IDEs is needed. We present and discuss a solution, motivated by a set of concrete examples: a domain specific aspect language for IDE events. In it, join points are events of interest that may trigger the advice in which the behavior of the IDE extension is called. We show how this allows for the development of IDE plugins and demonstrate the advantages over traditional publish/subscribe systems. [21]

AspectMaps: Extending Moose to visualize AOP software. When using aspect-oriented programming the application implicitly invokes the functionality contained in the aspects. Consequently program comprehension of such a software is more intricate. To alleviate this difficulty we developed the AspectMaps visualization and tool. AspectMaps extends the Moose program comprehension and reverse engineering platform with support for aspects, and is implemented using facilities provided by Moose. We present the AspectMaps tool, and show how it can be used by performing an exploration of a fairly large aspect-oriented application. We then show how we extended the FAMIX meta-model family that underpins Moose to also provide support for aspects. This extension is called ASPIX, and thanks to this enhancement Moose can now also treat aspect-oriented software. Finally, we report on our experiences using some of the tools in Moose; Mondrian to implement the visualization, and Glamour to build the user interface. We discuss how we were able to implement a sizable visualization tool using them and how we were able to deal with some of their limitations. [20]

6.3. Software Quality: Taming Software Evolution

APIEvolutionMiner: Keeping API Evolution under Control. During software evolution, source code is constantly refactored. In real-world migrations, many methods in the newer version are not present in the old version (e.g., 60% of the methods in Eclipse 2.0 were not in version 1.0). This requires changes to be consistently applied to reflect the new API and avoid further maintenance problems. We propose a tool to extract rules by monitoring API changes applied in source code during system evolution. In this process, changes are mined at revision level in code history. Our tool focuses on mining invocation changes to keep track of how they are evolving. We also provide three case studies in order to evaluate the tool. [34]

Towards an Automation of the Mutation Analysis Dedicated to Model Transformation. A major benefit of Model Driven Engineering (MDE) relies on the automatic generation of artefacts from high-level models through intermediary levels using model transformations. In such a process, the input must be well-designed and the model transformations should be trustworthy. Due to the specificities of models and transformations, classical software test techniques have to be adapted. Among these techniques, mutation analysis has been ported and a set of mutation operators has been defined. However, mutation analysis currently requires a considerable manual work and suffers from the test data set improvement activity. This activity is seen by testers as a difficult and time-consuming job, and reduces the benefits of the mutation analysis. This work addresses the test data set improvement activity. Model transformation traceability in conjunction with a model of mutation operators, and a dedicated algorithm allow to automatically or semi-automatically produce test models that detect new faults. The proposed approach is validated and illustrated in a case study written in Kermeta. [17]

Predicting software defects with causality tests. We propose a defect prediction approach centered on more robust evidences towards causality between source code metrics (as predictors) and the occurrence of defects. More specifically, we rely on the Granger causality test to evaluate whether past variations in source code metrics values can be used to forecast changes in time series of defects. Our approach triggers alarms when changes made to the source code of a target system have a high chance of producing defects. We evaluated our approach in several life stages of four Java-based systems. We reached an average precision greater than 50% in three out of the four systems we evaluated. Moreover, by comparing our approach with baselines that are not based on causality tests, it achieved a better precision. [19]

6.4. Software Quality: History and Changes

Tracking dependencies between code changes: An incremental approach. Merging a change often leads to the question of knowing what are the dependencies to other changes that should be merged too to obtain a working system. This question also arises with code history trackers – Code history trackers are tools that react to what the developer do by creating first-class objects that represent the change made to the system. We evaluate the capacity of different code history trackers to represent, also as first-class objects, the dependencies between those changes. We also present a representation for dependencies that works with the event model of Epicea, a fine-grained and incremental code history tracker. [32]

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 aheuristic-based approach for architecture conformance can be used to rapidly raise architectural warnings, without deeply involving experts in the process. [22]

6.5. Reconciling Dynamic Languages and Isolation

Delegation Proxies: The Power of Propagation. Scoping behavioral variations to dynamic extents is useful to support non-functional requirements that otherwise result in cross-cutting code. Unfortunately, such variations are difficult to achieve with traditional reflection or aspects. We show that with a modification of dynamic proxies, called delegation proxies, it becomes possible to reflectively implement variations that propagate to all objects accessed in the dynamic extent of a message send. We demonstrate our approach with examples of variations scoped to dynamic extents that help simplify code related to safety, reliability, and monitoring. [38]

Reifying the Reflectogram. Reflective facilities in OO languages are used both for implementing language extensions (such as AOP frameworks) and for supporting new programming tools and methodologies (such as object-centric debugging and message-based profiling). Yet controlling the run-time behavior of these reflective facilities introduces several challenges, such as computational overhead, the possibility of meta-recursion and an unclean separation of concerns between base and meta-level. We present five dimensions of meta-level control from related literature that try to remedy these problems. These dimensions are namely: temporal and spatial control, placement control, level control and identity control. We argue that the reification of the descriptive notion of the reflectogram, can unify the control of meta-level execution in all these five dimensions. We present a model for the reification of the reflectogram and validate our approach through a prototype implementation in the Pharo programming environment. Finally we detail a case study on run-time tracing illustrating our approach. [35]

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. [26]

6.6. Dynamic Languages: Virtual Machines

Benzo: Reflective Glue for Low-level Programming. The goal of high-level low-level programming is to bring the abstraction capabilities of high-level languages to the system programming domain, such as virtual machines (VMs) and language runtimes. However, existing solutions are bound to compilation time and expose limited possibilities to be changed at runtime and from language-side. They do not fit well with fully reflective languages and environments. We propose Benzo1, a lightweight framework for high-level low-level programming that allows developers to generate and execute at runtime low-level code. It promotes the implementation, and dynamic modification, of system components with high-level language tools outperforming

existing dynamic solutions. Since Benzo is a general framework we choose three applications that cover an important range of the spectrum of system programming for validating the infrastructure: a Foreign Function Interface (FFI), primitives instrumentation and a just-in-time bytecode compiler (JIT). With Benzo we show that these typical VM-level components are feasible as reflective language-side implementations. Due to its unique combination of high-level reflection and low-level programming, Benzo shows better performance for these three applications than the comparable high-level implementations. [30]

A bytecode set for adaptive optimizations. The Cog virtual machine features a bytecode interpreter and a baseline Just-in-time compiler. To reach the performance level of industrial quality virtual machines such as Java HotSpot, it needs to employ an adaptive inlining com-piler, a tool that on the fly aggressively optimizes frequently executed portions of code. We decided to implement such a tool as a bytecode to bytecode optimizer, implemented above the virtual machine, where it can be written and developed in Smalltalk. The optimizer we plan needs to extend the operations encoded in the bytecode set and its quality heavily depends on the bytecode set quality. The current bytecode set understood by the virtual machine is old and lacks any room to add new operations. We decided to implement a new bytecode set, which includes additional bytecodes that allow the Just-in-time compiler to generate less generic, and hence simpler and faster code sequences for frequently executed primitives. The new bytecode set includes traps for validating speculative inlining de-cisions and is extensible without compromising optimization opportunities. In addition, we took advantage of this work to solve limitations of the current bytecode set such as the maximum number of instance variable per class, or number of literals per method. We plan to have it in production in the Cog virtual machine and its Pharo, Squeak and Newspeak clients in the coming year. [43]

6.7. Traits

Trait-oriented Programming in Java 8 Java 8 was released recently. Along with lambda expressions, a new language construct is introduced: default methods in interfaces. The intent of this feature is to allow interfaces to be extended over time preserving backward compatibility. We show a possible, different use of interfaces with default methods: we introduce a trait-oriented programming style based on an interface-as-trait idea, with the aim of improving code modularity. Starting from the most common operators on traits, we introduce some programming patterns mimicking such operators and discuss this approach. [29]

6.8. Tailoring Applications

In the context of the PhD of G. Polito, we developed Tornado, a way to generate specialized and minimal runtime. Using a run-fail-grow approach, which tries to execute an expression in an empty world, and on failure copies the missing program elements from a mother environment to the currently empty world, we could grew 11k full reflective application adding two numbers or 18k for the 100 factorial expression. We also usd this approach to generate specialized webserver in around 500kb. These results show that we can generate hyperspecialized kernels.

7. Bilateral Contracts and Grants with Industry

7.1. 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.

7.2. 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 2014.

7.3. Resilience FUI

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

We proposed to use WebWorkers as a way to control DOM edition. 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. We proposed a digest of the semantics of JavaScript centered around security concerns.

7.4. 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.

7.5. Pharo Consortium

The Pharo Consortium was founded in 2012 and is growing constantly. As of end 2014, it has 14 company members, 10 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). Two co-supervisions are ongoing (Guillermo Polito, Max Mattone). 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 partnernship is done with the following members from the LIRMM-D'OC-APR: Marianne Huchard, Roland Ducournau, Jean-Claude König, Rodolphe 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).

The project started in march 2011 and ended this year in November just after the defense of PhD student André Hora

8.3. European Initiatives

8.3.1. 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 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

8.3.2. Collaborations in European Programs, except FP7 & H2020

8.3.2.1. 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.

8.4. International Initiatives

8.4.1. Inria International Labs

CIRIC Chile and Pleiad Team of University of Chile at Santiago

We are collaborating with ObjectProfile, a startup company which is hosted at Inria Chile. ObjectProfile is a collaborator within the PLOMO2 Associated Team and a contributor to both Pharo and Moose. http://objectprofile.com

The DeepIntoPharo book is a collaboration with the Pleiad Team of University of Chile at Santiago.

8.4.2. Inria Associate Teams

8.4.2.1. PLOMO2

Title: Infrastructure for a new generation of development tools

International Partner:

Universidad de Chile (Chile), DCC.

Duration: 2014 - 2016

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 kind of information supporting the task at hand. The goal of the first PLOMO was 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 maintenance). With Plomo2, we want to build on top of this work and invent a new generation of tools to navigate and profile programs.

The hypotheses that Plomo2 will seek to verify are:

- Use of reflection enables new profiling techniques

- Use of visualization in a programming environment improves programmer performance

The overall objectives of Plomo2 are:

- Infrastructure for profiling programs and recording programmer activity.
- Visual software maps defined in a flexible and agile fashion
- Combining dynamic information with visualization to improve the development environment
- Empirical evaluation of this environment
- All the efforts will be performed on Pharo and Moose, two platforms heavily used by the RMoD and Pleiad teams.

The detailed work plan and the results of the first year can be found in the PLOMO2 report at http://pleiad.cl/research/plomo2.

8.4.3. Inria International Partners

8.4.3.1. 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).

8.4.3.2. 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.

8.4.4. Participation In other International Programs

8.4.4.1. STIC AmSud

Participants: Damien Cassou [correspondant], Gustavo Santos [RMoD], Martin Martin [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.

8.4.4.2. 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 got a paper accepted to SANER 2015.

8.5. International Research Visitors

8.5.1. Visits of International Scientists

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

- Ronie Saldago: 24/08/2014 until 07/09/2014. Subject was FFI and OSWindow.
- Miguel Campusano: 16/08/2014 until 11/09/2014. Subject was Slots and visual representation of code.
- Alexandre Bergel: 13/12/2014 until 01/01/2015. Subject: system support for advanced profiling.
- Juraj Kubelka: 06/12/2014 until 19/12/2014. First visit to RMoD to plan future collaboration.

In the context of MEALS:

- Guido Chari visited RMoD from November 2014.

Other visitors:

- Laurence Tratt, Software Development Team, King's College London (15-16/05/14)
- Johan Fabry, University of Chile, November 2014.
- Max Leske, University of Bern, Mar 2014.
- Miao Fang, Siemens, from Jun 2014 until Jul 2014.
- Alain Plantec, Univ. Bretagne Occidentale, Jan 2014

8.5.1.1. Internships

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

Lucas Godoy, Universidad de Buenos Aires (Argentina):*Tracking depeendencies between code changes.*, May 2014 - Oct 2014.

Baptiste Quide, Polytech Lille: *Package dependencies analysis*, from May 2014 until Aug 2014.

Hayatou Oumarou, Universite de Maroua, Cameroun:*Cost of Rules*, frome June 2014 to Oct 2014.

Clara Allende, Universidad Tecnológica Nacional, Buenos Aires (Argentina): *BreakPoints for Pharo*, from May 2014 until Oct 2014.

Max Mattone, École des Mines Douai: *VirtualCPU for Pharo*. From May 2014 until Oct 2014.

Mark Rizun, Ivan Franko National University of Lviv, Ukraine: *Refactoring Improvements*, from July 2014 until Aug 2014.

Kevin Lanvin, University Lille: *A web front-end for Moose*, from Jan until Apr 2014.

Leo Perard, University Lille: *Telescope: a new way to describe visualizations*, from Mar 2014 until Aug 2014.

8.5.2. Visits to International Teams

- Stéphane Ducasse visited LAM Research, Inc, USA for one week in December 2014.
- Stéphane Ducasse visited the University of Delft, 3 days, July 2014
- Stefan Marr visited the Software Composition Group at Universität Bern in Switzerland for two days in December 2014
- Stefan Marr visited the Institut für Systemsoftware at the JKU University Linz in Austria for three days in July 2014
- Stefan Marr visited the Software Development Team of Laurence Tratt at King's College London for two days in May 2014
- Martín Dias visited the University of Technology of Delft for one week in September, 2014.
- Martín Dias visited the University of Buenos Aires in January 2014.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific events organisation

9.1.1.1. Member of the organizing committee

- Marcus Denker, Damien Cassou and Stéphane Ducasse where part of the organization of ESUG 2014.
- Damien Cassou co-organized Dyla'14 *Workshop on Dynamic Languages and Applications*

9.1.2. Scientific events selection

9.1.2.1. Member of the conference program committee

- Anne Etien
  - ECMFA 2014, *European Conference on Modelling Foundations and Applications*.
  - IWST 2014, *International Workshop on Smalltalk Technologies*.
  - ME 2014, *Models and Evolution*
  - SLE 2014, *International Conference on Software Language Engineering*.
  - Dyla'14, *Workshop on Dynamic Languages and Applications*
  - Inforsid 14 *Congrès INFORSID - INFormatique des ORganisations et Systèmes d'Information et de Décision*
- Damien Cassou
  - Dyla'14, *Workshop on Dynamic Languages and Applications*
- Stefan Marr
  - DLS 2014, *Dynamic Langauge Symposium*
  - 9th ICOOOLPS 2014 *Workshop Implementation, Compilation, Optimization of OO Languages, Programs and Systems*.
- Marcus Denker
  - IWST 2014, *International Workshop on Smalltalk Technologies*.
  - COP 2014, *ECOOP International Workshop on Context-Oriented Programming*
- Stéphane Ducasse declined the participation to ECOOP 2014 and ECOOP 2015.

9.1.2.2. Reviewer

- Damien Cassou: Apsec 2014 *The Asia-Pacific Software Engineering Conference*
- Martin Dias: ICSME 2014 *International Conference on Software Maintenance and Evolution*
- Andre Hora: ICSME 2014 and SANER 2014 *International Conference on Software Analysis, Evolution, and Reengineering*

9.1.3. Journal

9.1.3.1. Reviewer

- Anne Etien: International Journal of Computer Aided Engineering and Technology (IJCAET)
- Damien Cassou reviewed paper for SCP

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

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

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

Master : Damien Cassou, Conception des applications réparties, 42h, M1, Université Lille 1, France

Licence : Damien Cassou, UV Info, 24h, L3, Telecom Lille, France

Licence : Damien Cassou, Conception orientée objet, 57h, L3, Université Lille 1, France

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

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

Master : Anne Etien, Système d'information objet, 22h, M1, Polytech-Lille, France

Master : Anne Etien, Bases de données, 44h, M1, Polytech-Lille, France

Master : Anne Etien, Bases de données Avancées, 12h, M1, Polytech-Lille, France

Licence : Anne Etien, Bases de données, 22h, L3, Polytech-Lille, France

Licence : Camille Teruel, Conception orientée objet, 36h, L3, Université Lille 1, France

License : Damien Pollet, Programmation orientée objet, 85h, L3, Telecom Lille, France

License : Damien Pollet, Systèmes Numériques (SNUM), 11h, L3, Telecom Lille, France

Master : Damien Pollet, Ingénierie Logicielle (ILOG), 7h, M1, Telecom Lille, France

Master : Damien Pollet, Technologies de Systèmes d'Information Ouverts (TSIO), 38h, M1, Telecom Lille, France

Master : Damien Pollet, Spécification et Validation du Logiciel (SVL), 24h, M1, Université Lille 1, France

Master : Damien Pollet, Services Réseaux (SER), 28h, M2, Telecom Lille, France

Licence : Nicolas Anquetil, Interfaces et Conception Orientée Objets, 64h, L2, IUT-A/Université de Lille-1, France

Licence : Nicolas Anquetil, Analyse et conception des systèmes d'information, 24h, L2, IUT-A/Université de Lille-1, France

Licence : Nicolas Anquetil, stage, 6h, L2, IUT-A/Université de Lille-1, France

Licence : Nicolas Anquetil, Projet tutoré, 16h, L2, IUT-A/Université de Lille-1, France

Licence : Nicolas Anquetil, Conception et programmation objet avancées, 48h, L2, IUT-A/Université de Lille-1, France

Licence : Nicolas Anquetil, Projet Modélisation mathématique, 16h, L2, IUT-A/Université de Lille-1, France

Licence : Nicolas Anquetil, Introduction aux interfaces homme-machine, 32h, L2, IUT-A/Université de Lille-1, France

Licence : Jean-Christophe Bach, Conception orientée objet, 42h, L3, Université Lille 1, France

Master : Jean-Christophe Bach, Génie logiciel, 42h, M1, Université Lille 1, France

Licence : Stéphane Ducasse, Cours IAE (2014), 14h, Annecy, France

Master : Stéphane Ducasse, Cours Ecole des Mines de Douai (2014), 12h, France

Master : Stéphane Ducasse, Cours USTL L3 (2014), 4.5h, Université Lille, France

License : Stéphane Ducasse, Advanced Design, 6h, University of Delft, L3

9.2.2. Supervision

Camillo Bruni, Towards Self-aware Virtual Machines, 16/05/2014, Stéphane Ducasse, Marcus Denker.

Andre Hora, Assessing and Improving Rules to Support Software Evolution, 04/11/2014, 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 Fabresse, Stéphane Ducasse.

PhD in progress: Marco Naddeo, Viviana Bono (Univerosty of Turin), Damien Cassou, Stéphane Ducasse, started 01/01/2014

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: Max Mattone: Updating programs at runtime, started Oct 2014 Noury Bouraqadi, Luc Fabresse, Stéphane Ducasse .

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

9.2.3. Juries

- Nicolas Anquetil: Lisong Guo, LIP6 University Pierre and Marie Currie, 12/2014 (reviewer)
- Damien Cassou: Eyvind W. Axelsen, Oslo, 05/2014 (opponent)
- Stéphane Ducasse: *Une approche pragmatique pour mesurer la qualité des applications à base de composants logiciels*, Salma Hamza, Université de Bretagne Sud, France. 18/12/14. (referee)
- Anne Etien was in the mid term examination committee of the PhD thesis: Elie Richa 09/2014 (reviewer)

9.3. Popularization

- Marcus Denker gave two presentations about Pharo at the open source conference FOSDEM 2014.
- Damien Cassou gave a Pharo tutorial at PLDI'14.
- RIC Day was held 23 September 2014 @ University of Lille 1, Lille, Anne Etien and Damien Pollet gave a presentation about Pharo, Moose and software maintenance.
- A Programming Dojo was organized by Christophe Demarey on Dec 1st.
- A hands-on course about Pharo for the students at Lille in November organized by Léo Perard and Stéphane Ducasse.
- Demo for industry at the Rencontres Inria-Industrie *Technologies du Web*, 26 Nov 2014 at EuraTechnologies - Lille.
- RMOD co-organized and participated at ESUG 2014.
- Multiple public Pharo Sprints in Lille.
- Stefan Marr gave a half-day tutorial on "Building Self-Optimizing Interpreters with Truffle or RPython" at the Software Composition Group, Universität Bern, Switzerland, December 2nd, 2014
- Stefan Marr gave an invited talk on "Building High-Performance Language Implementations with Low Effort" at Software Composition Group, Universität Bern, Switzerland, December 1st, 2014
- Stefan Marr gave an invited talk on "Zero-Overhead Metaprogramming: Using Self-Optimizing Interpreters to Remove the Runtime Cost of Reflective Programming" at the Software Languages Lab, Vrije Universiteit Brussel, Belgium, November 2014
- Stefan Marr gave an invited talk "On the Way to Concurrency without Accidental Complexity" at the Institut für Systemsoftware, JKU University Linz, Austria, July 2014

- Stefan Marr gave an invited talk on “Building VMs for Language Designers” at the Software Development Team of Laurence Tratt, King’s College London, May 2014
- Stefan Marr gave a talk on “How to get a JIT Compiler for Free?” in the Smalltalk Developer Room of the FOSDEM Open Source Conference, Brussels, Belgium, February 2014

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° 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, 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] N. ANQUETIL. *Supporting Software Evolution in the Organizations*, universite de Liile-1, May 2014, Habilitation à diriger des recherches, https://hal.inria.fr/tel-01086785

[12] C. BRUNI. *Towards Self-aware Virtual Machines*, Université Lille 1 - Sciences et Technologies, May 2014, https://hal.inria.fr/tel-01097323

[13] A. CAVALCANTE HORA. *Assessing and Improving Rules to Support Software Evolution*, Université Lille 1 - Sciences et Technologies, November 2014, https://hal.inria.fr/tel-01087860

Articles in International Peer-Reviewed Journals

[14] H. ABDEEN, S. DUCASSE, D. POLLET, I. ALLOUI, J.-R. FALLERI. *The Package Blueprint: visually analyzing and quantifying package dependencies*, in "Science of Computer Programming", March 2014, vol. 89, n° Part C, pp. pp. 298-319 [*DOI :* 10.1016/J.SCICO.2014.02.016], https://hal.inria.fr/hal-00957695

[15] E. ALLENDE, O. CALLAU, J. FABRY, É. TANTER, M. DENKER. *Gradual Typing for Smalltalk*, in "Science of Computer Programming", December 2014, vol. 96, n° 1, pp. 52-69 [*DOI :* 10.1016/J.SCICO.2013.06.006], https://hal.inria.fr/hal-00862815

[16] V. ARANEGA, A. W. DE OLIVEIRA RODRIGUES, A. ETIEN, F. GUYOMARCH, J.-L. DEKEYSER. *Integrating Profiling into MDE Compilers*, in "International Journal of Software Engineering & Applications (IJSEA)", July 2014, vol. 5, n° 4, 20 p. [*DOI :* 10.5121/IJSEA.2014.5401], https://hal.inria.fr/hal-01053031

[17] V. ARANEGA, J.-M. MOTTU, A. ETIEN, T. DEGUEULE, B. BAUDRY, J.-L. DEKEYSER. *Towards an Automation of the Mutation Analysis Dedicated to Model Transformation*, in "Software Testing, Verification and Reliability", April 2014 [*DOI :* 10.1002/STVR.1532], https://hal.inria.fr/hal-00988164

[18] 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", September 2014, https://hal.inria.fr/hal-01060537

[19] C. COUTO, P. PIRES, M. TULIO VALENTE, R. BIGONHA, N. ANQUETIL. *Predicting software defects with causality tests*, in "Journal of Systems and Software", July 2014, vol. 93, pp. 24-41 [*DOI :* 10.1016/J.JSS.2014.01.033], https://hal.inria.fr/hal-01086783

[20] J. FABRY, A. KELLENS, S. DENIER, S. DUCASSE. *AspectMaps: Extending Moose to visualize AOP software*, in "Science of Computer Programming", 2014, vol. 79, pp. 6 - 22 [*DOI :* 10.1016/J.SCICO.2012.02.007], https://hal.inria.fr/hal-01086997

[21] J. FABRY, R. ROBBES, M. DENKER. *DIE: A Domain Specific Aspect Language for IDE Events*, in "J.UCS Journal of Universal Computer Science", February 2014, vol. 20, n^{o} 2, pp. 135-168, https://hal.inria.fr/hal-00936376

[22] 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

[23] S. MARR, T. PAPE, W. DE MEUTER. *Are We There Yet? Simple Language-Implementation Techniques for the 21st Century*, in "IEEE Software", September 2014, vol. 31, n^{o} 5, pp. 60-67 [*DOI :* 10.1109/MS.2014.98], https://hal.inria.fr/hal-01066793

[24] M. MARTINEZ PECK, N. BOURAQADI, L. FABRESSE, M. DENKER, C. TERUEL. *Ghost: A uniform and general-purpose proxy implementation*, in "Science of Computer Programming", December 2014, pp. 339-359, https://hal.inria.fr/hal-01081236

[25] 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

[26] 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, https://hal.inria.fr/hal-00903724

[27] V. UQUILLAS-GOMEZ, S. DUCASSE, A. KELLENS. *Supporting Streams of Changes during Branch Integration*, in "Science of Computer Programming", September 2014, https://hal.inria.fr/hal-01060534

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

International Conferences with Proceedings

[29] V. BONO, E. MENSA, M. NADDEO. *Trait-oriented Programming in Java 8*, in "PPPJ'14: International Conference on Principles and Practices of Programming on the Java Platform: virtual machines, languages, and tools", Cracow, Poland, September 2014, https://hal.inria.fr/hal-01026531

[30] C. BRUNI, S. DUCASSE, I. STASENKO, G. CHARI. *Benzo: Reflective Glue for Low-level Programming*, in "International Workshop on Smalltalk Technologies", Cambridge, United Kingdom, August 2014, https://hal.inria.fr/hal-01060551

[31] M. DE WAEL, S. MARR, T. VAN CUTSEM. *Fork/Join Parallelism in the Wild: Documenting Patterns and Anti-patterns in Java Programs Using the Fork/Join Framework*, in "PPPJ'14, International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools", Cracow, Poland, September 2014 [*DOI :* 10.1145/2647508.2647511], https://hal.inria.fr/hal-01064872

[32] C. DEMAREY, D. CASSOU, S. DUCASSE. *Towards a new package dependency model*, in "International Workshop on Smalltalk Technologies", Cambridge, United Kingdom, August 2014, https://hal.inria.fr/hal-01086083

[33] L. GODOY, D. CASSOU, S. DUCASSE. *Tracking dependencies between code changes: An incremental approach*, in "IWST 2014", Cambridge, United Kingdom, August 2014, https://hal.archives-ouvertes.fr/hal-01076238

[34] A. HORA, A. ETIEN, N. ANQUETIL, S. DUCASSE, M. T. VALENTE. *APIEvolutionMiner: Keeping API Evolution under Control*, in "Software Evolution Week (European Conference on Software Maintenance and Working Conference on Reverse Engineering)", Antwerp, Belgium, February 2014, https://hal.inria.fr/hal-00991722

[35] 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

[36] G. POLITO, N. BOURAQADI, S. DUCASSE, L. FABRESSE. *Understanding Pharo's global state to move programs through time and space*, in "International Workshop on Smalltalk Technologies", Edinburgh, United Kingdom, August 2014, https://hal.archives-ouvertes.fr/hal-01070964

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

[38] E. WERNLI, O. NIERSTRASZ, C. TERUEL, S. DUCASSE. *Delegation Proxies: The Power of Propagation*, in "Modularity", Lugano, Switzerland, April 2014, https://hal.inria.fr/hal-00958573

Research Reports

[39] M. DENKER, N. ANQUETIL, D. CASSOU, S. DUCASSE, A. ETIEN, D. POLLET. *Project-Team RMoD 2013 Activity Report*, Inria Lille, January 2014, https://hal.inria.fr/hal-00936375

[40] M. DIAS, V. UQUILLAS-GOMEZ, D. CASSOU, S. DUCASSE. *Software Integration Questions: A Quantitative Survey*, Inria Lille, December 2014, https://hal.inria.fr/hal-01093496

[41] G. POLITO, S. DUCASSE, N. BOURAQADI, L. FABRESSE. *Extended results of Tornado: A Run-Fail-Grow approach for Dynamic Application Tayloring*, May 2014, https://hal.archives-ouvertes.fr/hal-00996908

Other Publications

[42] N. ANQUETIL, S. DUCASSE, M. U. BHATTI. *Dedicated Software Analysis Tools*, October 2014, pp. 22-23, ERCIM news – Special theme Software Quality, https://hal.inria.fr/hal-01086593

[43] C. BÉRA, E. MIRANDA. *A bytecode set for adaptive optimizations*, August 2014, https://hal.inria.fr/hal-01088801

[44] N. PAPOULIAS, N. BOURAQADI, L. FABRESSE, S. DUCASSE, M. DENKER. *Mercury: a Model for Live Remote Debugging in Reflective Languages*, May 2014, https://hal.inria.fr/hal-00989294

References in notes

[45] 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

[46] 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

[47] 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

[48] 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

[49] 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

[50] 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

[51] 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

[52] 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

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

[54] 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

[55] 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

[56] 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

[57] 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

[58] 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

[59] 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

[60] 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

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

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

[63] A. GOLDBERG. *Smalltalk 80: the Interactive Programming Environment*, Addison WesleyReading, Mass., 1984

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

[65] 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

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

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

[68] 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

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

[70] 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

[71] 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

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

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

[74] 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

[75] 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

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

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

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

[79] 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

[80] 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

[81] 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

[82] 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

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

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

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

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

[87] 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