Project-Team RMoD (Analyses and Language Constructs for Object-Oriented Application Evolution) 2017 Activity Report

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

▶ **To cite this version:**

Marcus Denker, Nicolas Anquetil, Stéphane Ducasse, Anne Etien, Damien Pollet. Project-Team RMoD (Analyses and Language Constructs for Object-Oriented Application Evolution) 2017 Activity Report. [Research Report] INRIA Lille - Nord Europe. 2018. hal-01683649

HAL Id: hal-01683649
https://inria.hal.science/hal-01683649

Submitted on 14 Jan 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

Activity Report 2017

Project-Team RMOD

Analyses and Languages Constructs for Object-Oriented Application Evolution

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

RESEARCH CENTER
Lille - Nord Europe

THEME
Distributed programming and Software engineering

Table of contents

1. Personnel **1**
2. Overall Objectives **2**
2.1. Introduction 2
2.2. Reengineering and remodularization 2
2.3. Constructs for modular and isolating programming languages 3
3. Research Program **4**
3.1. Software Reengineering 4
3.1.1. Tools for understanding applications 4
3.1.2. Remodularization analyses 4
3.1.3. Software Quality 5
3.2. Language Constructs for Modular Design 5
3.2.1. Traits-based program reuse 5
3.2.2. Reconciling Dynamic Languages and Isolation 6
4. Application Domains **7**
4.1. Programming Languages and Tools 7
4.2. Software Reengineering 7
5. Highlights of the Year **7**
5.1.1. Release of Pharo 6 7
5.1.2. Pharo Consortium joins InriaSoft 7
5.1.3. Awards 7
5.1.4. Keynote at Programming 2017 7
6. New Software and Platforms **7**
6.1. Moose 7
6.2. Pharo 8
6.3. Pillar 8
7. New Results **9**
7.1. Software Quality: Testing and Tools 9
7.2. Software Reengineering 10
7.3. Dynamic Languages: Language Constructs for Modular Design 10
7.4. Dynamic Languages: Debugging 12
7.5. Dynamic Languages: Virtual Machines 13
7.6. Interaction 13
7.7. Software Engineering for BlockChain and Smart Contracts 13
8. Bilateral Contracts and Grants with Industry **14**
8.1. Bilateral Contracts with Industry 14
8.1.1. BlockChain 14
8.1.2. Pharo Consortium 14
8.2. Bilateral Grants with Industry 14
8.2.1. Worldline CIFRE 14
8.2.2. Thales CIFRE 14
8.2.3. Remodularization of Architecture 14
9. Partnerships and Cooperations **14**
9.1. Regional Initiatives 14
9.2. National Initiatives 15
9.3. European Initiatives 15
9.4. International Initiatives 16
9.5. International Research Visitors 16
9.5.1. Visits of International Scientists 16
9.5.2. Visits to International Teams 17

10. Dissemination **17**
10.1. Promoting Scientific Activities 17
10.1.1. Scientific Events Organisation 17
10.1.2. Scientific Events Selection 17
10.1.2.1. Chair of Conference Program Committees 17
10.1.2.2. Member of the Conference Program Committees 18
10.1.2.3. Reviewer 18
10.1.3. Journal 18
10.1.4. Invited Talks 18
10.1.5. Scientific Expertise 18
10.1.6. Research Administration 18
10.2. Teaching - Supervision - Juries 19
10.2.1. Teaching 19
10.2.2. Supervision 19
10.2.3. Juries 20
10.3. Popularization 21
11. Bibliography **21**

Project-Team RMOD

Creation of the Project-Team: 2009 July 01

Keywords:

Computer Science and Digital Science:

A2. - Software
A2.1. - Programming Languages
A2.1.2. - Object-oriented programming
A2.1.4. - Aspect-oriented programming
A2.1.9. - Dynamic languages
A2.1.10. - Domain-specific languages
A2.5. - Software engineering
A2.5.1. - Software Architecture & Design
A2.5.3. - Empirical Software Engineering
A2.5.4. - Software Maintenance & Evolution
A2.6. - Infrastructure software
A2.6.3. - Virtual machines

Other Research Topics and Application Domains:

B2. - Health
B2.7. - Medical devices
B5. - Industry of the future
B5.9. - Industrial maintenance
B6.5. - Information systems
B7. - Transport and logistics

1. Personnel

Research Scientists

Stéphane Ducasse [Team leader, Inria, Senior Researcher, HDR]
Andrew Black [Inria, Advanced Research Position, from Sep 2017 until Nov 2017]
Serge Demeyer [Inria, Advanced Research Position, from Sep 2017]
Marcus Denker [Inria, Researcher]

Faculty Members

Nicolas Anquetil [Univ des sciences et technologies de Lille, Associate Professor, HDR]
Anne Etien [Univ des sciences et technologies de Lille, Associate Professor]
Damien Pollet [Univ des sciences et technologies de Lille, Associate Professor]

Post-Doctoral Fellow

Henrique Santos Camargos Rocha [Inria, from May 2017]

PhD Students

Clément Béra [Inria, 3rd year, granted by Region Nord Pas de Calais, until Sep 2017]
Vincent Blondeau [Worldline, 3rd year, granted by CIFRE, until Oct 2017]
Gustavo Jansen de Souza Santos [Inria, 3rd year, granted by CNPq (Brazil), until Feb 2017]
Julien Delplanque [Univ des sciences et technologies de Lille, from Oct 2017]
Brice Govin [Thales, 3rd year, granted by CIFRE]

Jason Lecerf [CEA-LIST, 2nd year, co-supervision with Thierry Goubier]
Marco Naddeo [University of Torino, co-supervision with Pr. Viviana Bono, until Aug 2017]
Pablo Tesone [Inria-Ecole des Mines de Douai, 2nd year]
Thibault Raffaillac [Inria, 3rd year, co-supervision with Stéphane Huot of Mjolnir team Inria]

Technical staff

Clément Béra [Inria, from Oct 2017]
Pavel Krivanek [Inria]
Denis Kudriashov [Inria]
Esteban Lorenzano [Inria]
Christophe Demarey [Inria, Engineers, 70%]

Interns

Thomas Dupriez [Ecole Normale Supérieure Cachan, from Mar 2017]
Sophie Kaleba [Agence de services et de paiement, from Apr 2017 until Sep 2017]
Clement Mastin [Inria, from May 2017 until Aug 2017]
Amal Noussi Mbeyim [Ecole normale supérieure de Rennes, from May 2017 until Jul 2017]
Morgane Pigny [ECE Paris, until Feb 2017]
Jeremie Regnault [Inria, from Jun 2017 until Aug 2017]
Benoit Verhaeghe [Univ des sciences et technologies de Lille, from May 2017 until Aug 2017]

Administrative Assistant

Julie Jonas [Inria]

Visiting Scientists

Abdelghani Alidra [Badji Mokhtar University (Algeria), PhD Student, from May 2017 until Jun 2017]
Hamdi Gabsi [PhD student at National School of Computer Science Tunisia, from Oct 2017 until Nov 2017]
Markiyan Rizun [Student, from Aug 2017 until Sep 2017]
Moussa Saker [PhD Student Université Baji Mokhtar-Annaba (Algeria), from Dec 2017]
Ronie Salgado Faila [University of Chile, Sep 2017]

External Collaborators

Olivier Auverlot [Univ des sciences et technologies de Lille]
Guillermo Polito [CNRS]

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, Dynamic Software Update, Pharo, Moose.

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 focuses 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.

We seek to create tools to help developers perform large refactoring. How can they keep track of changes in various locations in a system while ensuring *integrity of current and new code* by *uniformally applying new design choices*.

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 [66]. 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 [63]. 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 [72]. 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.

[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.

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 [57], [56]. 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) [85] 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 [46]. 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 [73]. Until now, few works have attempted to identify layers in practice: Mudpie [87] is a first cut at identifying cycles between packages as well as package groups potentially representing layers. DSM (dependency structure matrix) [86], [81] 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 [69]. 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 [88], [60].

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 [83], [58] and classboxes [47] 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 [83], [58]. 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 [55], [77], [48], [59] and several type systems were defined [61], [84], [78], [71].

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 [50]. 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 [58], stateful [49], and freezable [59]) 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 [76]. 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 [54] then from Smalltalk [64].

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 [68]). 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 [51], as well as controlling the access to reflective features [52], [53] are important challenges. We plan to:

- Study the isolation abstractions available in erights (http://www.erights.org) [75], [74], and Java's class loader strategies [70], [65].
- Categorize the different reflective features of languages such as CLOS [67], Python and Smalltalk [79] and identify suitable isolation mechanisms and infrastructure [62].
- Assess different isolation models (access rights, capabilities [80]...) 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 [51],
  - the use of encapsulation policies as a basis to restrict the interfaces of the controlled objects [82],
  - 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...) [79].

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, http://consortium.pharo.org, has more than 25 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 analyzing software systems to support understanding and continuous development as well as software quality analysis.

5. Highlights of the Year

5.1. Highlights of the Year

5.1.1. Release of Pharo 6

We released a new version Pharo (Pharo 6). More information at http://pharo.org.

5.1.2. Pharo Consortium joins InriaSoft

The Pharo Consortium is joining InriaSoft (part of the Inria Foundation).

5.1.3. Awards

- Guillermo Polito, Luc Fabresse and Stéphane Ducasse won the 1st place in the best paper award at IWST 2017.
- Sophie Kaleba and Clément Béra won the 3rd place in the best paper awards at IWST 2017.
- Benoit Verhaeghe won the 2nd place for SmartTest in the Innovation Technologies Award at ESUG 2017.
- Denis Kudriashov with PharoThings won the 3rd place in the Innovation Technologies Award at ESUG 2017.

5.1.4. Keynote at Programming 2017

Stéphane Ducasse and Guillermo Polito did a keynote presentation in Modularity 2017, hosted within Programming 2017.

6. New Software and Platforms

6.1. Moose

Moose: Software and Data Analysis Platform
KEYWORDS: Software engineering - Meta model - Software visualisation

FUNCTIONAL DESCRIPTION: Moose is an extensive platform for software and data analysis. It offers multiple services ranging from importing and parsing data, to modeling, to measuring, querying, mining, and to building interactive and visual analysis tools. The development of Moose has been evaluated to 200 man/year.

Mots-cles : MetaModeling, Program Visualization, Software metrics, Code Duplication, Software analyses, Parsers

- Participants: Anne Etien, Nicolas Anquetil, Olivier Auverlot and Stéphane Ducasse
- Partners: Université de Berne - Sensus - Synectique - Pleiad - USI - Vrije Universiteit Brussel
- Contact: Stéphane Ducasse
- URL: http://www.moosetechnology.org

6.2. Pharo

KEYWORDS: Live programmation objet - Reflective system - Web Application
FUNCTIONAL DESCRIPTION: Pharo is a pure object reflective and dynamic language inspired by Smalltalk. In addition, Pharo comes with a full advanced programming environment developed under the MIT License. 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. Pharo 60 got 100 contributors world-wide. It is used by around 30 universities, 15 research groups and around 40 companies.

RELEASE FUNCTIONAL DESCRIPTION: Inspector/Playground/Spotter are new moldable development tools for inspecting, coding and searching objects. Slots model instance variables as first class entities and enable meta-programming on this level. ShoreLine reporter introduces a way to report system errors and collect statistics, that we will use for future improvements Dark theme.

- Participants: Christophe Demarey, Clement Bera, Damien Pollet, Esteban Lorenzano, Marcus Denker and Stéphane Ducasse
- Partners: Université de Berne - Cadence - Inceptive - Netstyle - Feenk - ObjectProfile - GemstoneSystems - Greyc Université de Caen - Basse-Normandie - BetaNine - Yesplan - RMod - Pleiad - Synectique - Sensus - Université de Bretagne Occidentale - École des Mines de Douai - Reveal
- Contact: Marcus Denker
- URL: http://www.pharo.org

6.3. Pillar

KEYWORDS: HTML - LaTeX - HTML5
FUNCTIONAL DESCRIPTION: Pillar is a markup syntax and associated tools to write and generate documentation and books. Pillar is currently used to write several books and other documentation. Two platforms have already been created on top of Pillar: PillarHub and Marina.

- Contact: Stéphane Ducasse
- URL: https://github.com/Pillar-markup/pillar

7. New Results

7.1. Software Quality: Testing and Tools

Testing Habits. What are the Testing Habits of Developers? We conducted a case study in a large IT company. Tests are considered important to ensure the good behavior of applications and improve their quality. But development in companies also involves tight schedules, old habits, less-trained developers, or practical difficulties such as creating a test database. As a result, good testing practices are not always used as often as one might wish. With a major IT company, we are engaged in a project to understand developers testing behavior, and whether it can be improved. Some ideas are to promote testing by reducing test session length, or by running automatically tests behind the scene and send warnings to developers about the failing ones. Reports on developers testing habits in the literature focus on highly distributed open-source projects, or involve students programmers. As such they might not apply to our industrial, closed source, context. We take inspiration from experiments of two papers of the literature to enhance our comprehension of the industrial environment. We report the results of a field study on how often the developers use tests in their daily practice, whether they make use of tests selection and why they do. Results are reinforced by interviews with developers involved in the study. The main findings are that test practice is in better shape than we expected; developers select tests ruthlessly (instead of launching an entire test suite); although they are not accurate in their selection, and; contrary to expectation, test selection is not influenced by the size of the test suite nor the duration of the tests. [23]

Tests in Open-Source. During the development, it is known that tests ensure the good behavior of applications and improve their quality. We studied developers testing behavior inside the Pharo community in the purpose to improve it. We report results of a field study on how often the developers use tests in their daily practice, whether they make use of tests selection and why they do. Results are strengthened by interviews with developers involved in the study. The main findings are that developers run tests every modifications of their code they did; most of the time they practice test selection (instead of launching an entire test suite); however they are not accurate in their selection; they change their selection depending on the duration of the tests and; contrary to expectation, test selection is not influenced by the size of the test suite. [35]

CodeCritics Applied to Database Schema: Challenges and First Results. Relational databases (DB) play a critical role in many information systems. For different reasons, their schemas gather not only tables and columns but also views, triggers or stored functions (i.e., fragments of code describing treatments). As for any other code-related artefact, software quality in a DB schema helps avoiding future bugs. However, few tools exist to analyze DB quality and prevent the introduction of technical debt. We present research issues related to assessing the software quality of a DB schema by adapting existing source code analysis research to database schemas. We present preliminary results that have been validated through the implementation of DBCritics, a prototype tool to perform static analysis on the SQL source code of a database schema. DBCritics addresses the limitations of existing DB quality tools based on an internal representation considering all entities of the database and their relationships. [26]

Recommending Source Code Locations for System Specific Transformations. From time to time, developers perform sequences of code transformations in a systematic and repetitive way. This may happen, for example, when introducing a design pattern in a legacy system: similar classes have to be introduced, containing similar methods that are called in a similar way. Automation of these sequences of transformations has been proposed in the literature to avoid errors due to their repetitive nature. However, developers still need support to identify all the relevant code locations that are candidate for transformation. Past research showed that these kinds of transformation can lag for years with forgotten instances popping out from time to time as other evolutions bring them into light. We evaluate three distinct code search approaches (structural, based on Information Retrieval, and AST based algorithm) to find code locations that would require similar transformations. We validate the resulting candidate locations from these approaches on real cases identified previously in literature. The results show that looking for code with similar roles, e.g., classes in the same hierarchy, provides interesting results with an average recall of 87% and in some cases the precision up to 70%. [33]

Quality-oriented Move Method Refactoring. Restructuring is an important activity to improve software internal structure. Even though there are many restructuring approaches, very few consider the refactoring impact on the software quality. In this paper, we propose an semi-automatic software restructuring approach based on quality attributes. We rely on the measurements of the Quality Model for Object Oriented Design (QMOOD) to recommend Move Method refactorings that improve software quality. In our preliminary evaluation on three open-source systems, our approach achieved an average recall of 57%. [34]

7.2. Software Reengineering

The Case for Non-Cohesive Packages. While the lack of cohesiveness of modules in procedural languages is a good way to identify modules with potential quality problems, we doubt that it is an adequate measure for packages in object-oriented systems. Indeed, mapping procedural metrics to object-oriented systems should take into account the building principles of object-oriented programming: inheritance and late binding. Inheritance offers the possibility to create packages by just extending classes with the necessary increment of behavior. Late binding coupled to the "Hollywood Principle" are a key to build frameworks and let the users branch their extensions in the framework. Therefore, a package extending a framework does not have to be cohesive, since it inherits the framework logic, which is encapsulated in framework packages. In such a case, the correct modularization of an extender application may imply low cohesion for some of the packages. We confirm these conjectures on various real systems (JHotdraw, Eclipse, JEdit, JFace) using or extending OO frameworks. We carry out a dependency analysis of packages to measure their relation with their framework. The results show that framework dependencies form a considerable portion of the overall package dependencies. This means that non-cohesive packages should not be considered systematically as packages of low quality. [22]

Identifying Classes in Legacy JavaScript Code. JavaScript is the most popular programming language for the Web. Although the language is prototype-based, developers can emulate class-based abstractions in JavaScript to master the increasing complexity of their applications. Identifying classes in legacy JavaScript code can support these developers at least in the following activities: (i) program comprehension; (ii) migration to the new JavaScript syntax that supports classes; and (iii) implementation of supporting tools, including IDEs with class-based views and reverse engineering tools. We propose a strategy to detect class-based abstractions in the source code of legacy JavaScript systems. We report on a large and in-depth study to understand how class emulation is employed, using a dataset of 918 JavaScript applications available on GitHub. We found that almost 70% of the JavaScript systems we study make some usage of classes. We also performed a field study with the main developers of 60 popular JavaScript systems in order to validate our findings. The overall results range from 97% to 100% for precision, from 70% to 89% for recall, and from 82% to 94% for F-score. [20]

A Critical Analysis of String APIs: The Case of Pharo. Most programming languages, besides C, provide a native abstraction for character strings, but string APIs vary widely in size, expressiveness, and subjective convenience across languages. In Pharo, while at first glance the API of the String class seems rich, it often feels cumbersome in practice; to improve its usability, we faced the challenge of assessing its design. However, we found hardly any guideline about design forces and how they structure the design space, and no comprehensive analysis of the expected string operations and their different variations. We first analyze the Pharo 4 String library, then contrast it with its Haskell, Java, Python, Ruby, and Rust counterparts. We harvest criteria to describe a string API, and reflect on features and design tensions. This analysis should help language designers in understanding the design space of strings, and will serve as a basis for a future redesign of the string library in Pharo. [19]

7.3. Dynamic Languages: Language Constructs for Modular Design

An Experiment with Lexically-bound Extension Methods for a Dynamic Language. An extension method is a method declared in a package other than the package of its host class. Thanks to extension methods, developers can adapt classes they do not own to their needs: adding methods to core classes is a typical use case. This is particularly useful for adapting software and therefore increasing reusability. In most

dynamically-typed languages, extension methods are globally visible. Because any developer can define extension methods for any class, naming conflicts occur: if two developers define an extension method with the same signature in the same class, only one extension method is visible and overwrites the other. Similarly, if two developers each define an extension method with the same name in a class hierarchy, one overrides the other. Existing solutions typically rely on a dedicated and slow method lookup algorithm to resolve conflicts at runtime. We present a model of scoped extension methods that minimizes accidental overrides and we present an implementation in Pharo that incurs little performance overhead. This implementation is based on lexical scope and hierarchy-first strategy for extension scoping. [44]

Scoped Extension Methods in Dynamically-Typed Languages. An extension method is a method declared in a package other than the package of its host class. Thanks to extension methods, developers can adapt to their needs classes they do not own: adding methods to core classes is a typical use case. This is particularly useful for adapting software and therefore to increase reusability. Inquiry. In most dynamically-typed languages, extension methods are globally visible. Because any developer can define extension methods for any class, naming conflicts occur: if two developers define an extension method with the same signature in the same class, only one extension method is visible and overwrites the other. Similarly, if two developers each define an extension method with the same name in a class hierarchy, one overrides the other. To avoid such *accidental overrides*, some dynamically-typed languages limit the visibility of an extension method to a particular scope. However, their semantics have not been fully described and compared. In addition, these solutions typically rely on a dedicated and slow method lookup algorithm to resolve conflicts at runtime. Approach. In this article, we present a formalization of the underlying models of Ruby refinements, Groovy categories, Classboxes, and Method Shelters that are scoping extension method solutions in dynamically-typed languages. Knowledge. Our formal framework allows us to objectively compare and analyze the shortcomings of the studied solutions and other different approaches such as MultiJava. In addition, language designers can use our formal framework to determine which mechanism has less risk of *accidental overrides*. Grounding. Our comparison and analysis of existing solutions is grounded because it is based on denotational semantics formalizations. Importance. Extension methods are widely used in programming languages that support them, especially dynamically-typed languages such as Pharo, Ruby or Python. However, without a carefully designed mechanism, this feature can cause insidious hidden bugs or can be voluntarily used to gain access to protected operations, violate encapsulation or break fundamental invariants. [17]

First-Class Undefined Classes for Pharo: From Alternative Designs to a Unified Practical Solution. Loading code inside a Pharo image is a daily concern for a Pharo developer. Nevertheless, several problems may arise at loading time that can prevent the code to load or even worse let the system in an inconsistent state. We focus on the problem of loading code that references a class that does not exist in the system. We discuss the different flavors of this problem, the limitations of the existing Undeclared mechanism and the heterogeneity of Pharo tools to solve it. Then, we propose an unified solution for Pharo that reifies Undefined Classes. Our model of Undefined Classes is the result of an objective selection among different alternatives. We then validate our solution through two cases studies: migrating old code and loading code with circular dependencies. We also present the integration of this solution into Pharo regarding the needed Meta-Object Protocol for Undefined Classes and the required modifications of existing tools. [30]

Run-Fail-Grow: Creating Tailored Object-Oriented Runtimes. Producing a small deployment version of an application is a challenge because static abstractions such as packages cannot anticipate the use of their parts at runtime. Thus, an application often occupies more memory than actually needed. Tailoring is one of the main solutions to this problem i.e., extracting used code units such as classes and methods of an application. However, existing tailoring techniques are mostly based on static type annotations. These techniques cannot efficiently tailor applications in all their extent (e.g., runtime object graphs and metadata) nor be used in the context of dynamically-typed languages. We propose a run-fail-grow technique to tailor applications using their runtime execution. Run-fail-grow launches (a) a reference application containing the original application to tailor and (b) a nurtured application containing only a seed with a minimal set of code units the user wants to ensure in the final application. The nurtured application is executed, failing when it founds missing objects, classes or methods. On failure, the necessary elements are installed into the nurtured application from the reference one, and the execution resumes. The nurtured application is executed until it finishes, or until the

developer explicitly finishes it, for example in the case of a web application. resulting in an object memory (i.e., a heap) with only objects, classes and methods required to execute the application. To validate our approach we implemented a tool based on Virtual Machine modifications, namely Tornado. Tornado succeeds to create very small memory footprint versions of applications e.g., a simple object-oriented heap of 11kb. We show how tailoring works on application code, base and third-party libraries even supporting human interaction with user G. interfaces. These experiments show memory savings ranging from 95% to 99%. [18]

7.4. Dynamic Languages: Debugging

Unanticipated Debugging with Dynamic Layers. To debug running software we need unanticipated adaptation capabilities, especially when systems cannot be stopped, updated and restarted. Adapting such programs at runtime is an extreme solution given the delicate live contexts the debugging activity takes place. We introduce the Dynamic Layer, a construct in which behavioral variations are gathered and activated as a whole set of adaptations. Dimensions of Dynamic Layers activation are reified to allow very fine definitions of layer scopes and a fine grained selection of adapted entities. An experimental implementation with the Pharo language is evaluated through a runtime adaptation example. [25]

New Generation Debuggers. Locating and fixing bugs is a well-known time consuming task. Advanced approaches such as object-centric or back-in-time debuggers have been proposed in the literature, still in many scenarios developers are left alone with primitive tools such as manual breakpoints and execution stepping. We explore several advanced on-line debugging techniques such as advanced breakpoints and on-line execution comparison, that could help developers solve complex debugging scenarios. We analyze the challenges and underlying mechanisms required by these techniques. We present some early but promising prototypes we built on the Pharo programming language. We finally identify future research paths by analyzing existing research and connecting it to the techniques we presented before. [27]

Debugging Cyber-Physical Systems. Cyber-Physical Systems (CPS) integrate sensors and actuators to collect data and control entities in the physical world. Debugging CPS systems is hard due to the time-sensitive nature of a distributed applications combined with the lack of control on the surrounding physical environment. This makes bugs in CPS systems hard to reproduce and thus to fix. In this context, on-line debugging techniques are helpful because the debugger is connected to the device when an exception or crash occurs. We report on our experiences on applying two different on-line debugging techniques for a CPS system: remote debugging using the Pharo remote debugger and our IDRA debugger. In contrast to traditional remote debugging, IDRA allows to on-line debug an application locally in another client machine by reproducing the runtime context where the bug manifested. Our qualitative evaluation shows that IDRA provides almost the same interaction capabilities than Pharo's remote debugger and is less intrusive when performing hot-modifications. Our benchmarks also show that IDRA is significantly faster than the Pharo remote debugger, although it increases the amount of data transferred over the network. [29]

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 runtime 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 then discuss how these dimensions interact with language semantics in class-based OO languages in terms of: scoping, inheritance and first-class entities. We argue that the reification of the descriptive notion of reflectogram can unify the control of meta-level execution in all these five dimensions while expressing properly the underlying language semantics. We present an extended model for the reification of the reflectogram based on our additional analysis and validate our approach through a new prototype implementation that relies on byte-code instrumentation. Finally, we illustrate our approach through a case study on runtime tracing. [16]

7.5. Dynamic Languages: Virtual Machines

VM Profiler. Code profiling enables a user to know where in an application or function the execution time is spent. The Pharo ecosystem offers several code profilers. However, most of the publicly available profilers (MessageTally, Spy, GadgetProfiler) largely ignore the activity carried out by the virtual machine, thus incurring inaccuracy in the gathered information and missing important information, such as the Just-in-time compiler activity. We describe the motivations and the latest improvements carried out in VMProfiler, a code execution pro-filer hooked into the virtual machine, that performs its analysis by monitoring the virtual machine execution. These improvements address some limitations related to assessing the activity of native functions (resulting from a Just-in-time compiler operation): as of now, VMProfiler provides more detailed profiling reports, showing for native code functions in which bytecode range the execution time is spent. [28]

Sista: Saving Optimized Code in Snapshots for Fast Start-Up. Modern virtual machines for object-oriented languages such as Java HotSpot, Javascript V8 or Python PyPy reach high performance through just-in-time compilation techniques, involving on-the-fly optimization and deoptimization of the executed code. These techniques require a warm-up time for the virtual machine to collect information about the code it executes to be able to generate highly optimized code. This warm-up time required before reaching peak performance can be considerable and problematic. We propose an approach, Sista (Speculative Inlining SmallTalk Architecture) to persist optimized code in a platform-independent representation as part of a snapshot. After explaining the overall approach, we show on a large set of benchmarks that the Sista virtual machine can reach peak performance almost immediately after start-up when using a snapshot where optimized code was persisted. [24]

7.6. Interaction

This work is done in collaboration with team Mjolnir.

Turning Function Calls Into Animations. Animated transitions are an integral part of modern interaction frameworks. With the increasing number of animation scenarios, they have grown in range of animatable features. Yet not all transitions can be smoothed: programming systems limit the flexibility of frameworks for animating new things, and force them to expose low-level details to programmers. We present an ongoing work to provide system-wide animation of objects, by introducing a delay operator. This operator turns setter function calls into animations. It offers a coherent way to express animations across frameworks, and facilitates the animation of new properties. [31]

7.7. Software Engineering for BlockChain and Smart Contracts

Solidity Parsing Using SmaCC: Challenges and Irregularities. Solidity is a language used to implement smart contracts on a blockchain platform. Since its initial conception in 2014, Solidity has evolved into one of the major languages for the Ethereum platform as well as other blockchain technologies. Due to its popularity, there are many tools specifically designed to handle smart contracts written in Solidity. However , there is a lack of tools for Pharo to handle Solidity contracts. Therefore, we implemented a parser using SmaCC to serve as a base for further developing Solidity support in Pharo. We describe the parser creation, the irregularities we found in the Solidity grammar specification, and common practices on how to adapt the grammar to an LR type parser. Our experiences with parsing the Solidity language using SmaCC may help other developers trying to convert similar grammars. [32]

SmartInspect: Smart Contract Inspection. Smart contracts are embedded procedures stored with the data they act upon. Debugging deployed Smart Contracts is a difficult task since once deployed, the code cannot be reexecuted and inspecting a simple attribute is not easily possible because data is encoded. In this technical report, we present SmartInspect to address the lack of inspectability of a deployed contract. Our solution analyses the contract state by using decompilation techniques and a mirror-based architecture to represent the object responsible for interpreting the contract state. SmartInspect allows developers and also end-users of a contract to better visualize and understand the contract stored state without needing to redeploy, nor develop any ad-hoc code. [43]

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

8.1.1. BlockChain

Participants: Henrique Rocha, Marcus Denker, Stéphane Ducasse
From 2016, ongoing.

We started a new collaboration with a local startup (UTOCAT) about tools and languages in the context of Blockchain systems. The collaboration started with a 2 month exploration phase involving an engineer at Inria Tech. A postdoc started in 2017.

8.1.2. Pharo Consortium

Participants: Esteban Lorenzano, Clément Béra, Marcus Denker, Stéphane Ducasse
From 2012, ongoing.

The Pharo Consortium was founded in 2012 and is growing constantly. By the end 2017, it has 27 company members, 14 academic partners. Inria supports the consortium with one full time engineer starting in 2011. In 2018, the Pharo Consortium will join InriaSoft.

More at http://consortium.pharo.org.

8.2. Bilateral Grants with Industry

8.2.1. Worldline CIFRE

Participants: Vincent Blondeau, Anne Etien, Nicolas Anquetil
From 2014 to 2017.

We are working on improving the testing behavior of the developers.

The PhD started in October 2014 and finished in 2017: Vincent Blondeau, *Test Selection Practices in a Large IT Company*, CIFRE WorldLine, November 8th, University Lille 1 (France),

8.2.2. Thales CIFRE

Participants: Brice Govin, Anne Etien, Nicolas Anquetil, Stéphane Ducasse
From 2015, ongoing.

We are working on large industrial project rearchitecturization. PhD in progress: Brice Govin, *Support to implement a rejuvenated software architecture in legacy software*. CIFRE Thale started Jan 2015.

8.2.3. Remodularization of Architecture

Participants: Anne Etien, Nicolas Anquetil, Stéphane Ducasse
From 2017, ongoing.

We started a new collaboration with the software editor Berger Levrault about software architecture remodularization. The collaboration started with an end study project exploring the architecture used in the company in order to later migrate from GWT to Angular JS since GWT will not be backward supported anymore in the next versions. An internship and a PhD CIFRE thesis will successively start in 2018.

9. Partnerships and Cooperations

9.1. Regional Initiatives

9.1.1. CAR IMT Douai

Participants: Pablo Tesone, Guillermo Polito, Marcus Denker, Stéphane Ducasse with: L. Fabresse and N. Bouraqadi (IMT Douai)

From 2009, ongoing.

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

Currently, Pablo Tesone is doing a PhD co-supervided by RMOD and Pr. L. Fabresse and N. Bouraqadi. We are preparing a collaboration in the Context of CPER Data in 2018.

9.2. National Initiatives

9.2.1. CEA List

Participants: Jason Lecerf, Stéphane Ducasse with T. Goubier (CEA List)
From 2016, ongoing.

Jason Lecerf started a shared PhD Oct 2016: *Reuse of code artifacts for embedded systems through refactoring*.

9.3. European Initiatives

9.3.1. Collaborations in European Programs, Except FP7 & H2020

Namur University, Belgium
Participants: Anne Etien, Nicolas Anquetil, Olivier Auverlot, Stéphane Ducasse.

From Sept 2016 to Dec. 2018.

Lille Nord Europe European Associated Team with the PreCISE research center of Pr. A. Cleve from Namur University (Belgium).

This project aims to study the co-evolution between database structure and programs and to propose recommendations to perform required changes on cascade. These programs are either internal to the schema as functions or triggers or external as applications written in Java or Php built on top of the DB. Our intuition is that software engineering techniques can be efficient for such issues. This project also aims to unify the abstract representation of the DB and its relationships with the internal or external program.

University of Turin (Italy)
Participants: Marco Naddéo, Stéphane Ducasse.
From 2015 to 2017.

Marco Naddéo was a PhD student co-supervised by Damien Cassou, Stéphane Ducasse at RMoD and Viviana Bono from University of Turin (Italy): *A modular Approach of Object initialization for Pharo*, University Turin, November 2017.

VUB Brussels, Belgium
Participants: Guillermo Polito, Stéphane Ducasse.

From 2016, ongoing.

Student: Matteo Marra, collaboration with Eliza Gonzalez Boix. Guillermo Polito co-supervised Matteo Marra's master thesis. This collaboration led to a workshop paper [29] and a paper under revision for Programming 2018.

University of Prague
Participants: Stéphane Ducasse.

From 2015, ongoing.

We are working with Dr. Robert Pergl from the University of Prague. Stéphane Ducasse gave a lecture at the University of Prague in 2017.

9.4. International Initiatives

9.4.1. Informal International Partners

Uqbar Argentina

Participants: Pablo Tesone, Esteban Lorenzano, Guillermo Polito, Stéphane Ducasse.

From 2015, ongoing.

We are working with the Uqbar team from different Argentinian universities. We hired three of the people: Nicolas Passerini(engineer), Esteban Lorenzano (engineer) and Pablo Tesone (PhD).

Pharo in Research:

Participants: Pablo Tesone, Esteban Lorenzano, Guillermo Polito, Marcus Denker, Stéphane Ducasse.
From 2009, ongoing.

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.

9.5. International Research Visitors

9.5.1. Visits of International Scientists

- Prof. Serge Demeyer, Universiteit Antwerpen, Belgium. September 1st until December 31st, 2017. Sabbatical on the topic of Test Automation
- Andrew Black, Department of Computer Science, Portland State University, September 1st until December 9th. Sabbatical, Implementing the Grace Language using Pharo
- Fernando Brito, Université de Lisbonne, 27 to 18/2/2017
- Sébastien Proksch, 09/02/2017
- Sébastien Martinez, Université de Rennes 1, 09/03/2017
- Sergiu Ivanov, CNRS Grenoble, 9 to 10/2/2017
- Coen de Roover, Université Bruxelles, 28/02/17
- Yoshiki Oshima, YCombinator Research, 13 to 17/3/17
- Abdelghani Alidra, Université de Skikda, 15/05 to 18/06/2017
- Sergiu Ivanov, CNRS Grenoble, 21/04/17
- Ronie Salgado, Université du Chili, 11/9 to 22/9/2017
- Andy Zaidman, Université de Delft, 08/11/17
- Laurence Tratt, King's College London, 15/09/17
- Elisa Gonzales, Université de Bruxelles, 15/09/17
- Théo D'Hondt, Université de Bruxelles, 15/09/17
- Rim Drira, RIADI Laboratory - National School of Computer Science, Tunisia, 09/11/17
- Gordana Rakic, Université de Belgrade - Serbie, 29/11 to 6/12/2017
- Henda Ben Gezahla, Ecole Nationale des Sciences de l'Informatique (ENSI) en Tunisie, 7/11 to 12/11/2017
- Abir Mbaya, Université de Lyon, 11 to 15/12/2017
- Olivier Flückiger (Northeastern University, US) 28/11 to 01/12. Talk: Correctness of Speculative Optimizations with Dynamic Deoptimization
- Gabriel Scherer (Parsifal, Inria Saclay, France) 28/11 to 29/11. Talk: Correctness of Speculative Optimizations with Dynamic Deoptimization

9.5.1.1. Internships

- Thomas Dupriez, ENS Cachan/Paris-Saclay, from 2017-03-16 until 2017-07-21, and from 2017-08-07 until 2017-08-11
- Sophie Kaleba, from Apr 2017 until Sep 2017
- Clement Mastin, from May 2017 until Aug 2017
- Amal Noussi Mbeyim, Ecole Normale Supérieure de Rennes, from May 2017 until Jul 2017
- Morgane Pigny, until Feb 2017
- Jeremie Regnault, from Jun 2017 until Aug 2017
- Benoit Verhaeghe, Université des Sciences et Technologies de Lille, from May 2017 until Aug 2017

9.5.2. Visits to International Teams

- Anne Etien: Labri, Université Bordeaux 1, January 2017.
- Nicolas Anquetil, Julien Delplanque and Anne Etien, Visit Namur University (Belgium), Decembre 2017.
- Stéphane Ducasse: Technical University Prague, Czech Republic.
- Stéphane Ducasse: ENIS Tunisia.
- Stéphane Ducasse: University of Novi Sad, Serbia.
- Stéphane Ducasse: Maribor, Slovenia.
- Stéphane Ducasse: VUB Bussels, Belgium.
- Stéphane Ducasse: University de Bretagne Occidentale.
- Stéphane Ducasse: Software Vomposition Group University of Bern/Switzerland.
- Guillermo Polito: VUB Brussel, Belgium.

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events Organisation

10.1.1.1. Member of the Organizing Committees

- Marcus Denker and Stéphane Ducasse are in the board of ESUG and organized ESUG 2017, the yearly Smalltalk conference that brings together research and industry http://www.esug.org/
- RMoD organized Pharo Days 2017, the yearly Pharo meeting of both industrial and research users
- Serge Demeyer served as organizing chair for the BENEVOL 2017 Workshop http://ansymore.uantwerpen.be/events/benevol2017

10.1.2. Scientific Events Selection

10.1.2.1. Chair of Conference Program Committees

- Anne Etien refused to be program chair of SCAM 2017 (International Working Conference on Source Code Analysis & Manipulation) because of financial matter, to keep the money for a PhD student to travel and presents his work
- Anne Etien was chair of the artifact evaluation track of VISSOFT 2017 http://vissoft17.dcc.uchile.cl/
- Anne Etien served as chair for IWST 2017
- Serge Demeyer served as chair for the industrial track of the POEM 2017 International Conference https://kuleuvencongres.be/poem2017/

10.1.2.2. Member of the Conference Program Committees

- Marcus Denker: SANER 2017 (23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering)
- Clément Béra: IWST 2017 (International Workshop on Smalltalk Technologies)
- Nicolas Anquetil SCAM 2017 (International Working Conference on Source Code Analysis & Manipulation)
- Nicolas Anquetil BENEVOL 2017 (BElgian-NEtherlands software eVOLution seminar)
- Anne Etien BENEVOL 2017 (BElgian-NEtherlands software eVOLution seminar)
- Anne Etien SATToSE 2017 (Seminar on Advanced Techniques & Tools for Software Evolution)
- Anne Etien ME 2017 (Workshop on Model Evolution)

10.1.2.3. Reviewer

- Thibault Raffaillac: IWST 2017 (International Workshop on Smalltalk Technologies)
- Thibault Raffaillac: ManLang 2017 (formerly PPPJ, Managed Languages and Runtimes)
- Clément Béra: ManLang 2017 (formerly PPPJ, Managed Languages and Runtimes)

10.1.3. Journal

10.1.3.1. Reviewer - Reviewing Activities

- Clément Béra: JUCS 2017 (Journal of Universal Computer Science)
- Nicolas Anquetil did Reviews for the following international journals:
  - TSE (IEEE Transactions on Software Engineering)
  - TOSEM (ACM Transactions on Software Engineering and Methodology)
  - IET Software
  - SCICO (Science of Computer Programming)
  - JISYS (Journal of Intelligent Systems)
  - JUCS (Journal of Universal Computer Science)

10.1.4. Invited Talks

- Stéphane Ducasse and Guillermo Polito did a keynote presentation in Modularity 2017, hosted within Programming 2017
- Nicolas Anquetil: Friday November, 10 2017, invited talk at LIRMM (Montpellier)

10.1.5. Scientific Expertise

Nicolas Anquetil: Grant reviewer for NSERC (Natural Sciences and Engineering Research Council of Canada)

10.1.6. Research Administration

Nicolas Anquetil: GDR/GPL (Groupement de Recherche Génie de la Programmation et du Logiciel), animation of the Rimel group (Rétro-Ingénierie, Maintenance et Evolution des Logiciels)

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

Licence: Nicolas Anquetil, Graphes et langages, 32h, L2, Univ. Lille 1, IUT-A, France

Licence: Nicolas Anquetil, Conception et programmation objet avancées, 28h, L2, Univ. Lille 1, IUT-A, France

Licence: Nicolas Anquetil, Principes des systèmes d'exploitation, 40h, L2, Univ. Lille 1, IUT-A, France

Licence: Nicolas Anquetil, Conception et développement d'applications mobiles, 30h, L2, Univ. Lille 1, IUT-A, France

Licence: Nicolas Anquetil, Méthodologie de la production d'applications, 33h, L2, Univ. Lille 1, IUT-A, France

Licence: Nicolas Anquetil, Modélisations mathématiques, 14h, L2, Univ. Lille 1, IUT-A, France

Licence: Nicolas Anquetil, Conception et programmation objet, 24h, L2, Univ. Lille 1, IUT-A, France

Licence: Stéphane Ducasse, OOP, 21h Prague technical University

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

Licence: Damien Pollet, Java, data structures and algorithms lecture, 50h, L3, Telecom Lille, France

Licence: Damien Pollet, Java lecture, 20h, L3, Telecom Lille, France

Licence: Damien Pollet, Java project, 20h, L3, Telecom Lille, France

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

License: Damien Pollet, Developer communities, 35h, L3, Université de Lille, France

Master: Anne Etien, Metamodelisation for Reverse Engineering, 25h, M2, Montpellier, France

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

Master: Anne Etien, Test et Maintenance, 14h, M2, Polytech-Lille, 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, 20h, M1, Polytech-Lille, France

Master: Anne Etien, Qualité logicielle, 8h, M2, University Lille 1, France

Master: Clément Béra, Software engineering, 30h, M2, Universite de Lille 1, France

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

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

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

Master: Stéphane Ducasse, Meta Programming, 12h, IMT Douai

Master: Stéphane Ducasse, Advanced Design, 24h, Catholic University of Lviv.

Master: Nicolas Anquetil, Qualité logicielle, 8h, M2, University Lille 1, France

Master: Nicolas Anquetil, Visualisation for Reverse Engineering, 6h, M2, Montpellier, France

Master: Thibault Raffaillac, ACT, 34h, M1, University Lille 1, France

Master: Christophe Demarey, Intégration Continue, 4h, M2, GIS2A5, Polytech-Lille, France

Master: Christophe Demarey, Intégration Continue, 12h, M2, IAGL, Univ. Lille 1, France

PhD: Stéphane Ducasse, Advanced Design, 24h, ENIS, Tunisia

10.2.2. Supervision

PhD: Gustavo Santos, *Assessing and Improving Code Transformations to Support Software Evolution*, Februrary 28th, Université Lille 1 (France), Anne Etien, Nicolas Anquetil

PhD: Vincent Blondeau, *Test Selection Practices in a Large IT Company*, CIFRE WorldLine, November 8th, Université Lille 1 (France), Anne Etien, Nicolas Anquetil

PhD: Marco Naddeo, *A modular Approach of Object initialization for Pharo*, Université de Turin, November 2017, Prof. V. Bono, Stéphane Ducasse

PhD: Clément Béra, *Sista: a Metacircular Architecture for Runtime Optimisation Persistence*, 9/2017, Université Lille 1 (France), Marcus Denker, Stéphane Ducasse

PhD in progress: Pablo Tesone, *Hot Software Update In Robotics Applications*, IMT Lille-Douai, started in Apr 2016, Luc Fabresse, Stéphane Ducasse

PhD in progress: Brice Govin, *Support to Implement a Rejuvenated Software Architecture in Legacy Software* CIFRE Thales, started Jan 2015, Anne Etien, Nicolas Anquetil, Stéphane Ducasse

PhD in progress: Jason Lecerf, *Reuse of Code Artifacts for Embedded Systems Through Refactoring*, started Oct 2016, CEA Thierry Goubier, Stéphane Ducasse

PhD in progress: Thibault Raffaillac, *Languages and System Infrastructure for Interaction*, started 2/11/2015, Shared PhD, S. Huot, Stéphane Ducasse

PhD in progress: Julien Delplanque, *Software Engineering Techniques Applied to Databases*, started Oct 2017, Anne Etien, Nicolas Anquetil

10.2.3. Juries

Stephane Ducasse: *Augmenting Type Inference with Lightweight Heuristics*, Nevena Lazarevic, University of Bern, Switzerland, 20/06/17.

Stephane Ducasse: *Étendre des interpréteurs par détournement, ou comment étendre des interpréteurs sans en modifier le code*, Florent Marchand de Kerchove, Université de Nantes, France. 18/11/16.

Serge Demeyer: Davy Landman, PhD thesis, *Reverse Engineering Source Code: Empirical Studies of Limitations and Opportunities*, October 5th 2017, Universiteit van Amsterdam (Netherlands)

Serge Demeyer: Robert Minelli, PhD thesis, *Interaction-Aware Development Environments — Recording, Mining, and Leveraging IDE Interactions to Analyze and Support the Development Flow*, November 13th 2017, Universita della Svizzera italiana (Switzerland)

Serge Demeyer: Vincent Blondeau, PhD thesis, *Test Selection Habits of Developers in a Large IT Company*, October 8th 2017, Inria Lille - Nord Europe (France)

Anne Etien: Xuan Sang Le, PhD thesis, *Software/FPGA Co-design for Edge-computing: Promoting Object Oriented Design*, May 2017, l'École des Mines de Douai, ENSTA, Brest (France)

Anne Etien: Maroua Hachicha PhD thesis, *Un modèle de prise de décision dynamique basé sur la performance des processus métiers collaboratifs*, April 3rd 2017, Université de Lyon 2 (France)

Anne Etien: Gustavo Santos, PhD thesis, *Assessing and Improving Code Transformations to Support Software Evolution*, Februrary 28th, Université Lille 1 (France)

Anne Etien: Vincent Blondeau, PhD thesis, *Test Selection Practices in a Large IT Company*, November 8th, Université Lille 1 (France)

Marcus Denker: PhD thesis, *Clément Béra, Sista: a Metacircular Architecture for Runtime Optimisation Persistence*, 9/2017, Université Lille 1 (France)

Nicolas Anquetil: Luís Ferreira da Silva, PhD thesis, *A Pattern-Based Approach to Scaffold the IT Infrastructure Design Process*, Dec. 20th, 2017, University Nova de Lisboa, Lisboa (Portugal)

Nicolas Anquetil: Gustavo Santos, PhD thesis, *Assessing and Improving Code Transformations to Support Software Evolution*, Februrary 28th, Université Lille 1 (France)

Nicolas Anquetil: Vincent Blondeau, PhD thesis, *Test Selection Practices in a Large IT Company*, November 8th, Université Lille 1 (France)

10.3. Popularization

- Article in Popular Magazine: Clément Béra and Olivier Auverlot. Au coeur de la VM Pharo. In GNU/Linux Magazine, numéro 210, décembre 2017.
- Book: Pharo by Example 5, Square Bracket Associates, 2017 [37].
  http://books.pharo.org/updated-pharo-by-example/
- RMOD co-organized and participated at ESUG 2017.
  http://www.esug.org/wiki/pier/Conferences/2017
- Multiple public Pharo Sprints in Lille.
  https://association.pharo.org/events
- A MOOC for Pharo is online (Stéphane Ducasse).
  http://mooc.pharo.org
- Thibault Raffaillac: Co-organized the hub at the IUT for the CCC contest.
  https://informatique.univ-lille1.fr/ccc/2017/contest/
- Thibault Raffaillac:Organizing weekly programming trainings (10 students per session).
- Julien Deplanque organized Advent Of Code 2017 for Pharo.
  https://github.com/juliendelplanque/AdventOfCode2017WithPharo
- Christophe Demarey gave a Talk at ncrafts about Pharo.
  http://bordeaux.ncrafts.io

11. Bibliography

Major publications by the team in recent years

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

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

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

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

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

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

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

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

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

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

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] V. BLONDEAU. *Test Selection Practices in a Large IT Company*, Université Lille 1 - Sciences et Technologies, November 2017, https://hal.inria.fr/tel-01661467

[12] C. BÉRA. *Sista: a Metacircular Architecture for Runtime Optimisation Persistence*, Université de Lille 1, September 2017, https://hal.inria.fr/tel-01634137

[13] M. NADDEO. *A Modular Approach to Object Initialization for Pharo*, Dipartimento di Informatica, Università degli Studi di Torino ; Inria Lille Nord Europe - Laboratoire CRIStAL - Université de Lille, November 2017, https://hal.inria.fr/tel-01651738

[14] G. J. DE SOUZA SANTOS. *Assessing and Improving Code Transformations to Support Software Evolution*, Université de Lille, February 2017, https://hal.inria.fr/tel-01545596

Articles in International Peer-Reviewed Journals

[15] N. MILOJKOVIĆ, C. BÉRA, M. GHAFARI, O. NIERSTRASZ. *Mining inline cache data to order inferred types in dynamic languages*, in "Science of Computer Programming", January 2018, pp. 1-17, forthcoming [*DOI :* 10.1016/J.SCICO.2017.11.003], https://hal.inria.fr/hal-01666541

[16] N. PAPOULIAS, M. DENKER, S. DUCASSE, L. FABRESSE. *End-User Abstractions for Meta-Control: Reifying the Reflectogram*, in "Science of Computer Programming", June 2017, https://hal.inria.fr/hal-01424787

[17] G. POLITO, S. DUCASSE, L. FABRESSE, C. TERUEL. *Scoped Extension Methods in Dynamically-Typed Languages*, in "The Art, Science, and Engineering of Programming", August 2017, vol. 2, n^{o} 1 [*DOI :* 2018/2/1], https://hal.archives-ouvertes.fr/hal-01609310

[18] G. POLITO, L. FABRESSE, N. BOURAQADI, S. DUCASSE. *Run-Fail-Grow: Creating Tailored Object-Oriented Runtimes.*, in "The Journal of Object Technology", 2017, vol. 16, n^{o} 3, pp. 1 - 36 [*DOI :* 10.5381/JOT.2017.16.3.A2], https://hal.archives-ouvertes.fr/hal-01609295

[19] D. POLLET, S. DUCASSE. *A critical analysis of string APIs: The case of Pharo*, in "Science of Computer Programming", November 2017, pp. 1-12, https://arxiv.org/abs/1711.10713 [*DOI :* 10.1016/J.SCICO.2017.11.005], https://hal.inria.fr/hal-01651250

[20] L. H. SILVA, M. T. VALENTE, A. BERGEL, N. ANQUETIL, A. ETIEN. *Identifying Classes in Legacy JavaScript Code*, in "Journal of Software: Evolution and Process", February 2017 [*DOI :* 10.1002/SMR.1864], https://hal.inria.fr/hal-01471905

International Conferences with Proceedings

[21] A. ALIDRA, M. SAKER, N. ANQUETIL, S. DUCASSE. *Identifying class name inconsistency in hierarchy: a first simple heuristic*, in "IWST 2017 - 12th International Workshop on Smalltalk Technologies", Maribor, Slovenia, IWST '17 Proceedings of the 12th edition of the International Workshop on Smalltalk Technologies, ACM, September 2017, pp. 14:1–14:8 [*DOI :* 10.1145/3139903.3139920], https://hal.inria.fr/hal-01663603

[22] N. ANQUETIL, M. U. BHATTI, S. DUCASSE, A. HORA, J. LAVAL. *The Case for Non-Cohesive Packages*, in "SQAMIA 2017 - 6th workshop on Software Quality Analysis, Monitoring, Improvement, and Applications", Belgrade, Serbia, September 2017 [*DOI :* 10.1145/0000000.0000000], https://hal.inria.fr/hal-01585703

[23] V. BLONDEAU, A. ETIEN, N. ANQUETIL, S. CRESSON, P. CROISY, S. DUCASSE. *What are the Testing Habits of Developers?: A Case Study in a Large IT Company*, in "International Conference on Software Evolution and Maintenance", Shanghai, China, September 2017, https://hal.inria.fr/hal-01571655

[24] C. BÉRA, E. MIRANDA, T. FELGENTREFF, M. DENKER, S. DUCASSE. *Sista: Saving Optimized Code in Snapshots for Fast Start-Up*, in "Proceedings of the 14th International Conference on Managed Languages and Runtimes", Prague, Czech Republic, ACM, September 2017, pp. 1 - 11 [*DOI :* 10.1145/3132190.3132201], https://hal.inria.fr/hal-01596321

[25] S. COSTIOU, M. KERBOEUF, M. DENKER, A. PLANTEC. *Unanticipated Debugging with Dynamic Layers*, in "LASSY 2017 Live Adaptation of Software SYstems", Brussels, Belgium, April 2017, 6 p. [*DOI :* 10.1145/3079368.3079391], http://hal.univ-brest.fr/hal-01591077

[26] J. DELPLANQUE, A. ETIEN, O. AUVERLOT, T. MENS, N. ANQUETIL, S. DUCASSE. *CodeCritics applied to database schema: Challenges and first results*, in "IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)", Klagenfurt, Austria, February 2017, pp. 432 - 436 [*DOI :* 10.1109/SANER.2017.7884648], https://hal.inria.fr/hal-01596247

[27] T. DUPRIEZ, G. POLITO, S. DUCASSE. *Analysis and exploration for new generation debuggers*, in "International Workshop on Smalltalk Technology IWST'17", Maribor, Slovenia, ACM, September 2017, pp. 5:1–5:6 [*DOI :* 10.1145/3139903.3139910], https://hal.archives-ouvertes.fr/hal-01585338

[28] S. KALEBA, C. BERA, A. BERGEL, S. DUCASSE. *A detailed VM profiler for the Cog VM*, in "International Workshop on Smalltalk Technology IWST'17", Maribor, Slovenia, IWST '17 Proceedings of the 12th edition of the International Workshop on Smalltalk Technologies, September 2017, n° Article No. 6, https://hal.archives-ouvertes.fr/hal-01585754

[29] M. MARRA, E. GONZALEZ BOIX, S. COSTIOU, M. KERBOEUF, A. PLANTEC, G. POLITO, S. DUCASSE. *Debugging Cyber-Physical Systems with Pharo: An Experience Report*, in "Proceedings of the 12th Edition of the International Workshop on Smalltalk Technologies", Maribor, Slovenia, ACM, September 2017, pp. 8:1–8:10 [*DOI :* 10.1145/3139903.3139913], https://hal.archives-ouvertes.fr/hal-01585349

[30] G. POLITO, S. DUCASSE, L. FABRESSE. *First-Class Undefined Classes for Pharo: From Alternative Designs to a Unified Practical Solution*, in "Proceedings of the 12th Edition of the International Workshop on Smalltalk Technologies", Maribor, Slovenia, ACM, September 2017, pp. 9:1–9:8 [*DOI :* 10.1145/3139903.3139914], https://hal.archives-ouvertes.fr/hal-01585305

[31] T. RAFFAILLAC, S. HUOT, S. DUCASSE. *Turning Function Calls Into Animations*, in "The 9th ACM SIGCHI Symposium on Engineering Interactive Computing Systems", Lisbon, Portugal, ACM, June 2017, 6 p. [*DOI :* 10.1145/3102113.3102134], https://hal.inria.fr/hal-01564116

[32] H. S. C. ROCHA, S. DUCASSE, M. DENKER, J. LECERF. *Solidity Parsing Using SmaCC: Challenges and Irregularities*, in "Proceedings of the 12th Edition of the International Workshop on Smalltalk Technologies", Maribor, Slovenia, ACM, September 2017, pp. 2:1–2:9 [*DOI :* 10.1145/3139903.3139906], https://hal.inria.fr/hal-01651858

[33] G. SANTOS, K. V. R. PAIXÃO, N. ANQUETIL, A. ETIEN, M. DE ALMEIDA, S. DUCASSE. *Recommending Source Code Locations for System Specific Transformations*, in "24th IEEE International Conference on Software Analysis, Evolution, and Reengineering", Klagenfurt, Austria, February 2017, https://hal.inria.fr/hal-01441790

[34] C. M. SOUZA COUTO, H. S. C. ROCHA, R. TERRA. *Quality-oriented Move Method Refactoring*, in "BENEVOL 2017 - 16th BElgian-NEtherlands software eVOLution symposium", Antwerp, Belgium, December 2017, pp. 1-5, https://hal.inria.fr/hal-01663666

[35] B. VERHAEGHE, V. BLONDEAU, N. ANQUETIL, S. DUCASSE. *Usage of Tests in an Open-Source Community: A Case Study with Pharo Developers*, in "roceedings of the 12th Edition of the International Workshop on Smalltalk Technologies", Maribor, Slovenia, ACM, September 2017, pp. 4:1–4:9 [*DOI :* 10.1145/3139903.3139909], https://hal.inria.fr/hal-01579106

Scientific Books (or Scientific Book chapters)

[36] J. BRANT, J. LECERF, T. GOUBIER, S. DUCASSE. *Smacc: a Compiler-Compiler*, The Pharo Booklet Collection, Pharo, September 2017, https://hal.inria.fr/hal-01612820

[37] S. DUCASSE, D. CHLOUPIS, N. HESS, D. ZAGIDULIN, A. P. BLACK, O. NIERSTRASZ, D. POLLET, D. CASSOU, M. DENKER. *Pharo by Example 5*, Lulu.com & Square Bracket Associates, January 2017, pp. 1-358, https://hal.inria.fr/hal-01659495

[38] S. DUCASSE, P. KENNY. *Scraping HTML with XPath*, published by the authors, October 2017, 26 p. , https://hal.inria.fr/hal-01612689

[39] S. DUCASSE, D. POLLET. *Learning Object-Oriented Programming, Design and TDD with Pharo*, published by the authors, October 2017, 239 p. , https://hal.inria.fr/hal-01612687

[40] J. FABRY, S. DUCASSE. *The Spec UI framework*, published by the authors, February 2017, 84 p. , https://hal.inria.fr/hal-01612690

[41] E. LORENZANO, S. DUCASSE, J. FABRY, N. HARTL. *Voyage: Persisting Objects in Document Databases*, Square Bracket Associates, May 2017, 46 p. , https://hal.inria.fr/hal-01612823

Research Reports

[42] N. ANQUETIL, M. DENKER, S. DUCASSE, A. ETIEN, D. POLLET. *Project-Team RMoD 2016 Activity Report*, Inria Lille - Nord Europe, January 2017, https://hal.inria.fr/hal-01444225

[43] S. BRAGAGNOLO, H. S. C. ROCHA, M. DENKER, S. DUCASSE. *SmartInspect: Smart Contract Inspection Technical Report*, Inria Lille, December 2017, https://hal.inria.fr/hal-01671196

[44] S. DUCASSE, L. FABRESSE, G. POLITO, C. TERUEL. *An Experiment with lexically-bound extension methods for a dynamic language*, Inria Lille - Nord Europe, March 2017, https://hal.inria.fr/hal-01483756

Other Publications

[45] B. GOVIN, N. ANQUETIL, A. ETIEN, S. DUCASSE, A. MONEGIER. *How Can We Help Software Rearchitecting Efforts? Study of an Industrial Case*, January 2017, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01451242

References in notes

[46] 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

[47] 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

[48] 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

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

[50] 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

[51] 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

[52] 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

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

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

[55] 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

[56] 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

[57] 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

[58] 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

[59] 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/Duca07bFreezableTrait.pdf

[60] 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

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

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

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

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

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

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

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

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

[69] 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

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

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

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

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

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

[75] 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

[76] 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

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

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

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

[80] 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

[81] 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

[82] 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

[83] 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

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

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

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

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

[88] 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