Project-Team RMoD 2020 Activity Report

Marcus Denker, Nicolas Anquetil, Vincent Aranega, Steven Costiou, Stéphane Ducasse, Anne Etien, Damien Pollet

RESEARCH CENTRE

Lille - Nord Europe

IN PARTNERSHIP WITH:

Université de Lille

2020 ACTIVITY REPORT

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

DOMAIN

Networks, Systems and Services, Distributed Computing

THEME

Distributed programming and Software engineering

Contents

Project-Team RMOD

Creation of the Project-Team: 2009 July 01

Keywords

Computer Sciences and Digital Sciences:

A1.3.3. – Blockchain

A2. – Software

A2.1. – Programming Languages

A2.1.3. – Object-oriented programming

A2.1.8. – Aspect-oriented programming

A2.1.10. – Domain-specific languages

A2.1.12. – Dynamic languages

A2.3.1. – Embedded systems

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 Team members, visitors, external collaborators

Research Scientists

- Stéphane Ducasse [Team leader, Inria, Senior Researcher, HDR]
- Steven Costiou [Inria, Researcher]
- Marcus Denker [Inria, Researcher]

Faculty Members

- Nicolas Anquetil [Université de Lille, Associate Professor, HDR]
- Vincent Aranega [Université de Lille, Associate Professor]
- Anne Etien [Université de Lille, Professor, HDR]
- Damien Pollet [Université de Lille, Associate Professor]

PhD Students

- Santiago Bragagnolo [Berger-Levrault, from May 2020]
- Julien Delplanque [Université de Lille, until Sep 2020]
- Thomas Dupriez [Université de Lille]
- Carolina Hernandez Phillips [Inria]
- Mahugnon Honore Houekpetodji [Cim]
- Pierre Misse-Chanabier [Inria]
- Theo Rogliano [Inria]
- Benoit Verhaeghe [Berger-Levrault]
- Maximilian Ignacio Willembrinck Santander [Inria, from Oct 2020]
- Oleksandr Zaitsev [Arolla SAS]

Technical Staff

- Santiago Bragagnolo [Inria, Engineer, until Apr 2020]
- Christophe Demarey [Inria, Engineer, 60%]
- Cyril Ferlicot-Delbecque [Inria, Engineer, until Sep 2020]
- Esteban Lorenzano [Inria, Engineer]
- Hernan Morales [Inria, Engineer, from Oct 2020]
- Allex Oliveira [Inria, Engineer, until Oct 2020]
- Ronie Salgado Faila [Inria, Engineer, from Mar 2020 until Nov 2020]
- Pablo Tesone [Inria, Engineer]
- Clotilde Toullec [Inria, Engineer, from Jul 2020]

Interns and Apprentices

- Chia-Ling Bragagnolo [Afpa, until Mar 2020]
- Eric Brandwein [Inria, from Apr 2020 until Sep 2020]
- Laurine Dargaud [Inria, from Feb 2020 until Jul 2020]
- Theophile Heurlier [Inria, from Jul 2020 until Aug 2020]
- Sebastian Jordan [Inria, from Jun 2020 until Nov 2020]
- Philippe Lesueur [Inria, from Apr 2020 until Aug 2020]
- Nicolas Margulies [Ecole normale supérieure Paris-Saclay, from Jul 2020 until Aug 2020]
- Myroslava Romaniuk [Inria, until May 2020]
- Clotilde Toullec [Université de Lille, from Mar 2020 until Jun 2020]
- Esteban Villalobos Diaz [Inria, until Mar 2020]

Administrative Assistants

- Aurore Dalle [Inria, from Mar 2020]
- Julie Jonas [Inria, until Feb 2020]

Visiting Scientists

- Giuseppe Antonio Pierro [University of Cagliari], Italy]
- Gordana Rakic [Université de Novi Sad - Serbie, until Feb 2020]
- Moussa Saker [University Badji Mokhtar-Annaba, until Oct 2020, Algeria]

External Collaborators

- Luc Fabresse [École des Mines de Douai]
- Matteo Marra [VUB Brussels]
- Guillermo Polito [CNRS]

2 Overall objectives

2.1 Introduction

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 [50]. 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 [47]. 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 [56]. 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

[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.

remodularizing a meta-level architecture in the context of the design of an isolating dynamic language. Indeed at the extreme, modules and structural control of reflective features are the first steps towards flexible, dynamic, yet isolating, languages. As a result, we expect to demonstrate that having adequate composable units and scoping units will help the evolution and recomposition of an application.

3 Research program

3.1 Software Reengineering

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

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

To help us answer these questions, we work on enriching Moose, our reengineering environment, with a new set of analyses [41, 40]. 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) [69] 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 [30]. 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 [57]. Until now, few works have attempted to identify layers in practice: Mudpie [71] is a first cut at identifying cycles between packages as well as package groups potentially representing layers. DSM (dependency structure matrix) [70, 65] 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 [53]. 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 [72, 44].

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 [67, 42] and classboxes [31] 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 [67, 42]. 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 [39, 61, 32, 43] and several type systems were defined [45, 68, 62, 55].

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 [34]. 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 [42], stateful [33], and freezable [43]) 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 [60]. 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 [38] then from Smalltalk [48].

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 [52]). 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 [35], as well as controlling the access to reflective features [37, 36] are important challenges. We plan to:

- Study the isolation abstractions available in erights (`http://www.erights.org`) [59, 58], and Java's class loader strategies [54, 49].

- Categorize the different reflective features of languages such as CLOS [51], Python and Smalltalk [63] and identify suitable isolation mechanisms and infrastructure [46].

- Assess different isolation models (access rights, capabilities [64],...) 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 [35],

- the use of encapsulation policies as a basis to restrict the interfaces of the controlled objects [66],
- 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,...) [63].

4 Application domains

4.1 Programming Languages and Tools

Many of the results of RMoD are improving programming languages or development tools for such languages. As such the application domain of these results is as varied as the use of programming languages in general. Pharo, the language that RMoD develops, is used for a very broad range of applications. From pure research experiments to real world industrial use (the Pharo Consortium has 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 Social and environmental responsibility

5.1 Footprint of research activities

The main environmental footprint of RMoD is related to international travel. Meeting researchers in person is indispensable for research.

We try to reduce travel by using online meetings as much as possible. The team tries to reduce impact of daily local travel by the use of local transport and biking to work.

5.2 Impact of research results

Our work on language runtimes has potential impact to reduce energy consumption.

Reengineering can be understood as a kind of *"recycling"*. Our tools allow companies to use systems for a longer time, reducing environmental impact of software that is created as a new project.

All software we develop as part of our research is released as Open Source, all our publications are available in the HAL archive.

6 Highlights of the year

6.1 Awards

- Most Influential Paper Award: Models 2020 (ACM/IEEE International Conference on Model Driven Engineering Languages and Systems), for the article: "A model-driven traceability framework for

software product lines". One of two articles selected from 10 years of the Journal of Software and Systems Modeling.
https://hal.inria.fr/hal-00668175

- Anne Etien became professor at University of Lille.
- We released Pharo 8. More information at http://pharo.org.

7 New software and platforms

7.1 New software

7.1.1 Moose

Name: 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

URL: http://www.moosetechnology.org

Contact: Stéphane Ducasse

Participants: Anne Etien, Nicolas Anquetil, Stéphane Ducasse, Julien Delplanque, Cyril Ferlicot-Delbecque

Parners: Université de Berne, Sensus, Pleiad, USI, Vrije Universiteit Brussel

7.1.2 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.

URL: http://www.pharo.org

Contacts: Stéphane Ducasse, Marcus Denker

Participants: Christophe Demarey, Damien Pollet, Esteban Lorenzano, Marcus Denker, Stéphane Ducasse, Guillermo Polito

Parners: BetaNine, Reveal, Inceptive, Netstyle, Feenk, ObjectProfile, GemTalk Systems, Greyc Université de Caen - Basse-Normandie, Université de Berne, Yesplan, RMod, Pleiad, Sensus, Université de Bretagne Occidentale, École des Mines de Douai, ENSTA, Uqbar foundation Argentina, LAM Research, ZWEIDENKER, LifeWare, JPMorgan Chase, KnowRoaming, ENIT, Spesenfuchs, FINWorks, Esug, FAST, Ingenieubüro Schmidt, Projector Software, HRWorks, Inspired.org, Palantir Solutions, High Octane, Soops, Osoco, Ta Mère SCRL, University of Yaounde 1, Software Quality Laboratory, University of Novi Sad, Software Institute Università della Svizzera italiana, Universdad Nacional de Quilmes, UMMISCO IRD, Université technique de Prague

7.1.3 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. It is used in the tools developed by Feenk.com.

URL: https://github.com/Pillar-markup/pillar

Contact: Stéphane Ducasse

Parner: Feenk

8 New results

8.1 Dynamic Languages: Language Features

Sub-method, partial behavioral reflection with Reflectivity: Looking back on 10 years of use. Refining or altering existing behavior is the daily work of every developer, but that cannot be always anticipated, and software sometimes cannot be stopped. In such cases, unanticipated adaptation of running systems is of interest for many scenarios, ranging from functional upgrades to on-the-fly debugging or monitoring of critical applications. Inquiry. A way of altering software at run time is using behavioral reflection, which is particularly well-suited for unanticipated adaptation of real-world systems. Partial behavioral reflection is not a new idea, and for years many efforts have been made to propose a practical way of expressing it. All these efforts resulted in practical solutions, but which introduced a semantic gap between the code that requires adaptation and the expression of the partial behavior. For example, in Aspect-Oriented Programming, a pointcut description is expressed in another language, which introduces a new distance between the behavior expression (the Advice) and the source code in itself. Ten years ago, the idea of closing the gap between the code and the expression of the partial behavior led to the implementation of the Reflectivity framework. Using Reflectivity, developers annotate Abstract Syntax Tree (AST) nodes with meta-behavior which is taken into account by the compiler to produce behavioral variations. We present Reflectivity, its API, its implementation and its usage in Pharo. We reflect on ten years of use of Reflectivity, and show how it has been used as a basic building block of many innovative ideas. [5]

A new modular implementation for Stateful Traits. The term traits is overloaded in the literature. In this work we refer to traits as the stateless model and implementation described before [42]. Traits provide a flexible way to support multiple inheritance code reuse in the context of a single inheritance language. The Pharo programming language includes the second implementation of stateless traits based on the original version of Schaerli's one. Even if it is the second iteration of such an implementation, it presents several limitations. First, it does not support state in traits. Second, its implementation is monolithic i.e., it is deeply coupled with the rest of the language kernel: it cannot be loaded nor unloaded. Furthermore, trait support impacts all classes, even classes not using traits. In addition, while the development tools include full support to work with classes, trait support is more limited because classes and traits do not present the same Metaobject Protocol (MOP). Finally, being monolithic and integrated in the language kernel, it is difficult to extend this current implementation. This article describes a new modular and extensible implementation of traits: it is easily loadable and unloadable as any other package. In addition, classes not using traits are not impacted. Finally, this new implementation includes a new and carefully designed Metaobject Protocol (MOP) that is compatible with both classes and traits. This allows one to reuse existing tools as they do not require special support for traits. Then, following the semantics proposed for stateful traits in [33], we present a new implementation of stateful traits. This implementation is an extension of our new modular implementation. We implemented modular traits using specialized metaclasses as our main language extension mechanism. By replacing the implementation we reduced the Pharo Language Kernel size by 15%. This model and implementation are used in production since Pharo7.0 (January 2019). [8]

8.2 Dynamic Languages: Live Programming

Molecule: live prototyping with component-oriented programming. At Thales Defense Mission Systems, software products first go through an industrial prototyping phase. Prototyping are serious applications we experiment with our end-users during workshops. End-users have a central role in the design process of our products. They often ask for software modifications during demonstrations to experiment new ideas or to focus the existing design on their needs. We present how we combine Smalltalk's live-programming capabilities with software component models to obtain flexible and modular software designs in our context of live prototyping. We present Molecule, a Trait-based Lightweight Corba Component Model implementation in Pharo. Molecule components are standard Pharo classes using exclusively Traits to become software components. We benefit from the dynamic run-time modification capabilities of Pharo during demonstrations with our end-users, where we explore software designs in a lively way. [26]

Preserving Instance State during Refactorings in Live Environments. An important activity of software evolution consists in applying refactorings to enhance the quality of the code without changing its behaviour. Having a proper refactoring tool is a must-to in any professional development environment. In addition, live programming allows faster development than the usual edit-compile-debug process. During live programming sessions, the developer can directly manipulate instances and modify the state of the running program. However, when a complex refactoring is performed, instances may be corrupted (i.e., their state is lost). For example, when pushing an instance variable to a superclass there is a moment where the superclass does not have yet acquired the new instance variable and the subclass does not have it any more. It means that the value assigned to this instance variable in existing instances is lost after the refactoring. This problem is not anecdotal since 36% of the refactorings described in Fowler's catalogue corrupt instances when used in a live programming context. There is a need to manually migrate, regenerate or reload instances from persistent sources. This manual fix lowers the usefulness of live programming. In this context of live programming, we propose, AtomicRefactoring, a new solution based on Dynamic Software Update to preserve the state of the application after performing refactorings. We provide a working extension to the existing refactoring tool developed for the language Pharo (a new offspring inheriting from Smalltalk), allowing application developers to perform complex refactorings preserving the live state of the running program. [9]

8.3 Dynamic Languages: Debugging

Handling Error-Handling Errors: dealing with debugger bugs in Pharo. In Pharo, errors happening during the opening of a debugger provoke error-handling errors. The Pharo system then drops into a rudimentary emergency evaluator, which provides extremely limited debugging features. This is a real problem while developing debuggers, when debuggers are more subject to bugs. In addition, the Pharo debugging infrastructure exposes an heterogeneous, obscure interface with various usages and users. Therefore, trying to extend this infrastructure to cope with debuggers bugs is tedious. We present Oups, an improved debugger infrastructure for Pharo. Oups provides a unified interface as a single entry point to request the opening of debuggers. Upon a debugger opening request, Oups uses interchangeable debugger opening strategies to select which debugger to open. We implemented a strategy that allows for the debugging of a failing debugger by other debuggers instead of the emergency evaluator. Oups improves the resilience of the Pharo system for specific cases of error-handling errors that we analyse. [22]

First Infrastructure and Experimentation in Echo-debugging. As applications get developed, bugs inevitably get introduced. Often, it is unclear why a given code change introduced a given bug. To find this causal relation and more effectively debug, developers can leverage the existence of a previous version of the code, without the bug. But traditional debug-ging tools are not designed for this type of work, making this operation tedious. In this article, we propose as exploratory work the echo-debugger, a tool to debug two different executions in parallel, and the Convergence Divergence Mapping (CDM) algorithm to locate all the control-flow divergences and convergences of these executions. In this exploratory work, we present the architecture of the tool and a scenario to solve a non trivial bug. [25]

Object Miners: Acquire, Capture and Replay Objects to Track Elusive Bugs. Elusive bugs are difficult to observe and to reproduce. They are often caused by non-deterministic or unexpected events, inputs or computations. To track elusive bugs, it is necessary to narrow down the scope of the bug investigation. This helps reproducing and observing very specific aspects of the program's state and behavior. In object-oriented programs, it often comes down to finding and debugging specific objects. However, some objects are particularly hard to find. Typical hard cases are finding a temporary object or a single particular instance of a given class. This is the case when debugging UI elements, or programs with non-deterministic state. This capability of identifying objects of interest is crucial to the debugging of objects. Yet, debuggers addressing this problem only provide manual or limited ways to find such objects. We present Object Miners: a non-intrusive object-centric debugging approach for acquiring, capturing and replaying objects. We show how the miners acquire objects at run time from the sub-results of an instrumented expression. Miners capture objects with their execution context and replay them to freeze strategic parts of the execution, eliminating non-determinism. We present a Pharo implementation of Object Miners along with a performance and memory overhead evaluation. We present a debugger built on top of Object Miners, and demonstrate through a series of examples how the application of Object Miners facilitates the tracking of elusive bugs. These examples include the fixing of a non-deterministic bug in an IOT application, tooling support for object-centric debugging, and the tracking of a bug in a real-world program. [6]

Framework-aware debugging with stack tailoring. Debugging applications that execute within a framework is not always easy: the call-stack offered to developers is often a mix-up of stack frames that belong to different frameworks, introducing an unnecessary noise that prevents developers from focusing on the debugging task. Moreover, relevant application code is not always available in the call-stack because it may have already returned, or is available in another thread. In such cases, manually gathering all relevant information from these different sources is not only cumbersome but also costly. We introduce Sarto, a call-stack instrumentation layer that allows developers to tailor the stack to make debugging framework-aware. The goal is to improve the quality and amount information present in the call-stack to reduce debugging time without impacting the execution time. Sarto proposes a set of six stack operations that combined hide irrelevant information, introduce missing information, and relate dispersed debugging sources before this is fed to the debugger. [17]

8.4 Software Reengineering

Recommendations for Evolving Relational Databases. Relational databases play a central role in many information systems. Their schemas contain structural and behavioral entity descriptions. Databases must continuously be adapted to new requirements of a world in constant change while: (1) relational database management systems (RDBMS) do not allow inconsistencies in the schema; (2) stored procedure bodies are not meta-described in RDBMS such as PostgreSQL that consider their bodies as plain text. As a consequence, evaluating the impact of an evolution of the database schema is cumbersome, being essentially manual. We present a semi-automatic approach based on recommendations that can be compiled into a SQL patch fulfilling RDBMS constraints. To support recommendations, we designed a meta-model for relational databases easing computation of change impact. We performed an experiment to validate the approach by reproducing a real evolution on a database. The results of our experiment show that our approach can set the database in the same state as the one produced by the manual evolution in 75% less time. [13]

Challenges for Layout Validation: Lessons Learned. Companies are migrating their software systems. The migration process contemplates many steps, UI migration is one of them. To validate the UI migration, most existing approaches rely on visual structure (DOM) comparison. However, in previous work, we experimented such validation and reported that it is not sufficient to ensure a result that is equivalent or even identical to the visual structure of the interface to be migrated. Indeed, two similar DOM may be rendered completely differently. So, we decide to focus on the layout migration validation. We propose a first visual comparison approach for migrated layout validation and experiment it on an industrial case. Hence, from this first experiment and already existing studies on image comparison field, we highlight

challenges for layout comparison. For each challenge, we propose possible solutions, and we detail the three main features we need to create a good layout validation approach. [12]

From Business Process to Cloud Application. Business Process (BP) development can be defined as the process of constructing a workflow application by composing a set of services performing BP's activities. In this respect, Cloud Services (CSs) are being increasingly used in BP development to ensure a high level of performance with a low operating cost. Although large companies may benefit from CSs' advantages, Small and Medium-sized Enterprises (SMEs) and startups are falling behind in cloud usage due to missing Information Technology competence, (IT-competence). The crucial challenge facing SMEs and startups in cloud-based BP development is to effectively address the so-called business and IT alignment issue. It represents the alignment between two different domains; one that entails technical cloud resource requirements and another comprising business-level. Formerly, we present this issue as a discovery challenge of suitable CSs performing abstract BP's activities. To address this challenge, firstly, we introduce the concept of cloud-aware BP by proposing a Domain-Specific Language (DSL) named "BP4Cloud" to enrich BP modeling and cover both business and technical requirements. Secondly, we propose an Activity-Services Matching algorithm that automates the discovery of CSs performing BP's activities. As a part of the evaluation, we set up by clarifying the specification of BP4Cloud elements through a proof of concept implementation applied on a real BP. Then, we proceed by evaluating the precision and recall of our Activity-Service Matching algorithm. [15]

Modular Moose: A new generation software reverse engineering environment. Advanced reverse engineering tools are required to cope with the complexity of software systems and the specific requirements of numerous different tasks (re-architecturing, migration, evolution). Consequently, reverse engineering tools should adapt to a wide range of situations. Yet, because they require a large infrastructure investment, being able to reuse these tools is key. Moose is a reverse engineering environment answering these requirements. While Moose started as a research project 20 years ago, it is also used in industrial projects, exposing itself to all these difficulties. We present ModMoose, the new version of Moose. ModMoose revolves around a new meta-model, modular and extensible; a new toolset of generic tools (query module, visualization engine, ...); and an open architecture supporting the synchronization and interaction of tools per task. With ModMoose, tool developers can develop specific meta-models by reusing existing elementary concepts, and dedicated reverse engineering tools that can interact with the existing ones. [10]

Analysing Microsoft Access Projects: Building a model in a Partially Observable Domain. Due to the technology evolution, every IT Company migrates their software systems at least once. Reengineering tools build system models which are used for running software analysis. These models are traditionally built from source code analysis and information accessible by data extractors (that we call such information observable). In this article we present the case of Microsoft Access projects and how this kind of project is partially observable due to proprietary storing formats. We propose a novel approach for building models that allows us to overcome this problem by reverse engineering the development environment runtime through the usage of Microsoft COM interface. We validate our approach and implementation by fully replicating 10 projects, 8 of them industrial, based only on our model information. We measure the repli-cation performance by measuring the errors during the process and completeness of the product. We measure the replication error, by tracking replication operations. We used the scope and completeness measure to enact this error. Completeness is measured by the instrumentation of a simple and scoped diff based on a third source of information. We present extensive results and interpretations. We discuss the threats to validity, the possibility of other approaches and the technological restrictions of our solution. [11]

Suggesting Descriptive Method Names: An Exploratory Study of Two Machine Learning Approaches. Programming is a form of communication between the person who is writing code and the one reading it. Nevertheless, very often developers neglect readability, and even well-written code becomes less understandable as software evolves. Together with the growing complexity of software systems, this creates an increasing need for automated tools for improving the readability of source code. In this

work, we focus on method names and study how a descriptive name can be automatically generated from a method's body. We experiment with two approaches from the field of text summarization: One based on TF-IDF and the other on deep recurrent neural network. We collect a dataset of methods from 50 real world projects. We evaluate our approaches by comparing the generated names to the actual ones and report the result using Precision and Recall metrics. For TF-IDF, we get results as good as 28% precision and 45% recall; and for deep neural network, 46% precision and 32% recall. [19]

Characterizing Pharo Code: A Technical Report. Pharo is a modern dynamically-typed reflective pure object-oriented language. It is inspired from Smalltalk. Its unconventional syntax mimics natural language: arguments are not grouped around parentheses at the end but within the message, making expressions looks like sentences. In addition, all control flow operations are expressed as messages and the programmer can freely define new ones and as such define Domain Specific Languages for his task. In this technical report we discuss the statistical properties of source code that people write using Pharo programming language. We present the methodology and tooling for analysing source code selected from the projects of Pharo ecosystem. By analysing 50 projects, consisting of 824 packages, 13,935 classes, and 151,717 methods, we answer questions such as "what is a typical method length?" or "what percent of source code are literals?". [28]

8.5 BlockChain and Smart Data

An Organized Repository of Ethereum Smart Contracts' Source Codes and Metrics. Many empirical software engineering studies show that there is a need for repositories where source codes are acquired, filtered and classified. During the last few years, Ethereum block explorer services have emerged as a popular project to explore and search for Ethereum blockchain data such as transactions, addresses, tokens, smart contracts' source codes, prices and other activities taking place on the Ethereum blockchain. Despite the availability of this kind of service, retrieving specific information useful to empirical software engineering studies, such as the study of smart contracts' software metrics, might require many subtasks, such as searching for specific transactions in a block, parsing files in HTML format, and filtering the smart contracts to remove duplicated code or unused smart contracts. We afford this problem by creating Smart Corpus, a corpus of smart contracts in an organized, reasoned and up-to-date repository where Solidity source code and other metadata about Ethereum smart contracts can easily and systematically be retrieved. We present Smart Corpus's design and its initial implementation, and we show how the data set of smart contracts' source codes in a variety of programming languages can be queried and processed to get useful information on smart contracts and their software metrics. Smart Corpus aims to create a smart-contract repository where smart-contract data (source code, application binary interface (ABI) and byte code) are freely and immediately available and are classified based on the main software metrics identified in the scientific literature. Smart contracts' source codes have been validated by EtherScan, and each contract comes with its own associated software metrics as computed by the freely available software PASO. Moreover, Smart Corpus can be easily extended as the number of new smart contracts increases day by day. [7]

Towards a Smart Data Processing and Storage Model. In several domains it is crucial to store and manipulate data whose origin needs to be completely traceable to guarantee the consistency, trustworthiness and reliability on the data itself typically for ethical and legal reasons. It is also important to guarantee that such properties are also carried further when such data is composed and processed into new data. In this article we present the main requirements and theorethical problems that arise by the design of a system supporting data with such capabilities. We present an architecture for implementing a system as well as a prototype developed in Pharo. [27]

8.6 Other Results

An Interdisciplinary Model for Graphical Representation. The paper questions whether data-driven and problem-driven models are sufficient for a software to automatically represent a meaningful graphical representation of scientific findings. The paper presents descriptive and prescriptive case studies to

understand the benefits and the shortcomings of existing models that aim to provide graphical representations of data-sets. First, the paper considers data-sets coming from the field of software metrics and shows that existing models can provide the expected outcomes for descriptive scientific studies. Second, the paper presents data-sets coming from the field of human mobility and sustainable development, and shows that a more comprehensive model is needed in the case of prescriptive scientific fields requiring interdisciplinary research. Finally, an interdisciplinary problem-driven model is proposed to guide the software users, and specifically scientists, to produce meaningful graphical representation of research findings. The proposal is indeed based not only on a data-driven and/or problem-driven model but also on the different knowledge domains and scientific aims of the experts, who can provide the information needed for a higher-order structure of the data, supporting the graphical representation output. [18]

15 years of reuse experience in evolutionary prototyping for the defense industry. At Thales Defense Mission Systems, software products first go through an industrial prototyping phase. We elaborate evolutionary prototypes which implement complete business behavior and fulfill functional requirements. We elaborate and evolve our solutions directly with end-users who act as stake-holders in the products' design. Prototypes also serve as models for the final products development. Because software products in the defense industry are developed over many years, this prototyping phase is crucial. Therefore, reusing software is a high-stakes issue in our activities. Component-oriented development helps us to foster reuse throughout the life cycle of our products. The work stems from 15 years of experience in developing prototypes for the defense industry. We directly reuse component implementations to build new prototypes from existing ones. We reuse component interfaces transparently in multiple prototypes, whatever the underlying implementation solutions. This kind of reuse spans prototypes and final products which are deployed on different execution platforms. We reuse non-component legacy software that we integrate in our component architectures. In this case, we seamlessly augment standard classes with component behavior, while preserving legacy code. We present our component programming framework with a focus on component reuse in the context of evolutionary prototyping. We report three scenarios of reuse that we encounter regularly in our prototyping activity. [16]

9 Bilateral contracts and grants with industry

9.1 Bilateral contracts with industry

Pharo Consortium
Participants: Esteban Lorenzano, Marcus Denker, Stéphane Ducasse
From 2012, ongoing.
The Pharo Consortium was founded in 2012 and is growing constantly. By the end 2019, it has 25 company members, 19 academic partners. Inria supports the consortium with one full time engineer starting in 2011. In 2018, the Pharo Consortium joined InriaSoft.
http://consortium.pharo.org.

Siemens AG, Germany
Participants: Stéphane Ducasse, Anne Etien, Nicolas Anquetil
The Siemens Digital Industry Division approached our team to help them restructure a large legacy systems. The joined work resulted in a publication in 2019: Decomposing God Classes at Siemens.

Thales DMS, Brest
Participants: Steven Costiou, from 2020
Industrial R&D collaboration with Dr. Éric Le Pors, lead prototyping architect at Thales DMS (Brest). We work on 1) unanticipated object-centric debugging of HMI prototypes 2) we study the practices of Thales with software component reuse and its impact on their development process.

9.2 Bilateral grants with industry

Berger-Levrault: Remodularization of Architecture
Participants: Nicolas Anquetil, Santiago Bragagnolo, Stéphane Ducasse, Anne Etien, Benoît Verhaeghe
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 since GWT will not be backward supported anymore in the next versions. A PhD CIFRE thesis started in 2019: Benoît Verhaeghe, Support a l'automatisation de la migration d'interface d'applications Web : le cas de GWT vers Angular. Santiago Bragagnol started a CIFRE in 2020.

Arolla: Machine Learning-Based Recommenders to Support Software Evolution and Visualisations for Legacy Systems
Participants: Nicolas Anquetil, Stéphane Ducasse, Anne Etien, Oleksandr Zaitsev, Nour Jihene Agouf
We are collaborating with the council company, Arolla, about software evolution. Arolla has daily problems with identifying architecture, design, and deviations from those artefacts. The goal of Oleksandr's CIFRE (started in 2019) thesis is to experiment with different machine learning techniques that can help us automate the process of library migration. A new CIFRE PhD (from 2021) is based around the study of visualisation techniques that can help us understand legacy systems.

CIM, France
Participants: Honore Mahugnon Houekpetodji, Stéphane Ducasse, Nicolas Anquetil
Cifre thesis started: *Analyse multi-facettes et operationnelle pour la transformation des systèmes d'information.*

Lifeware, Switzerland
In collaboration with the Pharo Consortium, we improve Pharo. The goal is to be able to work with very large systems (>100K classes, millions of methods).

10 Partnerships and cooperations

10.1 International initiatives

10.1.1 Inria Associate Team

SADPC

Title: *Systems Analyses and Debugging for Program Comprehension*

Duration: 2020 - 2023

Coordinator: Stéphane Ducasse

Partners:

- Department of Electrical Engineering, Concordia University (Canada)
- Christopher Fuhrman : Ecole de Technologie Supérieure (Montreal)
- Fabio Petrillo, UQAC, Universitée du Québec à Chicoutimi
- Foutse Khomh, Polytechnique Montréal.

Inria contact: Stéphane Ducasse

Summary: Systemic changes in the past decades have pushed software systems into all aspects of our lives, from our homes to our cars to our factories. These systems, both legacy (e.g., handling contracts for the Dept of Defense of the USA since 1958) and very recent (e.g., running the latest smart

factory in France in 2019), are difficult to understand by software engineers because of their intrinsic complexity. These engineers need help understanding the systems they must adapt to the new requirements of our time. The proposed associate team considers three research directions to support the software engineers maintaining and evolving large software systems: (a) system analyses and (b) debugging for (c) program comprehension. (a) Complex algorithms often act or are perceived by software engineers as black boxes because of their intrinsic and accidental complexity, both in architecture, design, and implementation. We will develop new software analyses to support algorithm understanding. (b) Previous debugging techniques assume a unique software engineer performing a solitary debugging session. We will work on a language allowing software engineers to build their own debuggers to fit their collaborative debugging strategies. (c) Previous work on program comprehension proposed views to address one single problem at a given moment of the comprehension process. They only provide a subset of the information required by software engineers. We want to propose an approach to adapt and combine views using metadata.

10.1.2 Inria international partners

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 2018, the next lecture is planned for 2021.

University of Novi Sad, Serbia
Participants: Stéphane Ducasse, Anne Etien, Nicolas Anquetil, Vincent Aranega, Prof. G. Rakic
A collaboration with the University of Novi Sad, Serbia, started in 2018 with the University joining the Pharo Consortium as an academic member. A Master thesis has been co-supervised. We are both part of the COST action CERCIRAS.

University of Cagliari, Italy
Participants: Stéphane Ducasse, Prof R. Tonelli
Giuseppe Antonio Pierro is doing a PhD between the two teams. We are working on software engineering problems in the context of blockchain based software.

University of Chicoutimi (UCAQ) and Concordia University, Quebec
Participants: Steven Costiou, Dupriez and Stéphane Ducasse, from 2020
Collaboration with Pr. Fabio Petrillo (UCAQ) and Pr. Yann-Gaël Guéhéneuc (Concordia) on the empirical evaluation of variable breakpoints.

University of Zurich, Suisse
Participants: Steven Costiou, Stéphane Ducasse, from 2020
Collaboration with Pr. Alberto Bacchelli on the empirical evaluation of object-centric debugging operators.

Instituto Federal de Educação Ciência e Tecnologia do Ceará, Brazil
Parcipants: Vincent Aranega, from 2020
Collaboration with Pr. Antonio Wendell de Oliveira Rodrigues on smart language and DSL for live coding semi-autonomous physical systems (drones).

University of Catholica de Cochabamba, Bolivia
Participants: Stéphane Ducasse, Prof. Juan-Pablo Sandoval
Code transformation rules (License of Sebastian Jordan) and revisiting refactorings support.

10.2 International research visitors

10.2.1 Visits of international scientists

- Professor Antonio Wendell de Oliveira Rodrigues (Invited Professor via Université de Lille, remotely due to the COVID)
- Gordana Rakic [University of Novi Sad, Serbia, from Nov 2019 until Feb 2020]
- Moussa Saker [University Badji Mokhtar-Annaba, Algeria, until Oct 2020, partly remote due to Covid]
- Giuseppe Antonio Pierro [University of Cagliari]

10.2.2 Visits to international teams

Due to Covid, no visits were possible in 2020.

10.3 European initiatives

10.3.1 FP7 & H2020 Projects

RMOD is part of the COST project CERCIRAS: Connecting Education and Research Communities for an Innovative Resource Aware Society.
https://www.cost.eu/actions/CA19135

10.4 National initiatives

CAR IMT Douai
Participants: Pablo Tesone, Guillermo Polito, Marcus Denker, Stéphane Ducasse with: 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.

10.5 Regional initiatives

CPER DATA 2: Smart Data
Participants: Marcus Denker, Stéphane Ducasse, Ronnie Salgado
Funding to work on exploring advanced data models integrating provisions for traceability, revocation and ownership.

CPER DATA 1: PharoIoT
Participants: Marcus Denker, Stéphane Ducasse, Allex Oliveira with: L. Fabresse and N. Bouraqadi (IMT Douai)
From 2018 to 2020.
Funding to work on the PharoIoT platform in collaboration with IMT Douai. Allex Oliveira has started to explore to create a startup with this technology.

11 Dissemination

11.1 Promoting scientific activities

11.1.1 Scientific events: organisation

M. Denker and S. Ducasse are organisers of the ESUG international conference and summer school. It was cancelled in 2020.
http://www.esug.org

Member of the organizing committees

- Stéphane Ducasse was part of the Organizing Committee of IWBOSE 2020, the 3rd International Workshop on Emerging Trends in Software Engineering for Blockchain
  http://www.agile-group.org/iwbose2020/

Chair of conference program committees

- Vincent Aranega was chair of IWST 2020
- Stéphane Ducasse was PC chair of ICSR 2020 (International Conference of Software Reuse)
- Nicolas Anquetil was PC chair for the Software Evolution Track at Quatic 2020

Member of the conference program committees

- Anne Etien was in the Program committee of: ICSR 2020, IWST 2020, Sattose 2020, ICSME 2020, ICSOFT 2020, ICSE 2021, ICPC 2020
- Nicolas Anquetil: PC member for SCAM 2020
- Vincent Aranega: PC member of IWST 2020, ICSOFT 2020, MODELSWARD 2020
- Steven Costiou: PC member IWST 2020, Défis GDR-GPL 2020

Reviewer

- Steven Costiou: Reviewer for ICPC 2020 and ICSR 2020
- Vincent Aranega: Reviewer for ICSME and SOSYM special Issue

11.1.2 Leadership within the scientific community

- GT GLIA working group of the CNRS GDR GPL: Anne Etien is co-leader of the GT GLIA working group of the Cnrs GDR-GPL (Software Engineering and artificial intelligence)
- GT Debugging working group of the CNRS GDR GPL: Steven Costiou is leader of the GT Debugging working group of the CNRS GDR GPL.

11.1.3 Scientific expertise

- Nicolas Anquetil: expert for the French government on CIR Research Tax Credit.
- Anne Etien: expert for the French government on CIR Research Tax Credit.
- Stéphane Ducasse: expert for the French government on CIR Research Tax Credit.
- Anne Etien: expert for the ANRT to evaluate CIFRE PhD proposal.

11.1.4 Research administration

- Anne Etien animates the thematic group of Software Engineering and is member of scientific council of CRIStAL lab.
- Anne Etien is elected member of the center committee of Inria Lille Nord Europe center.

11.2 Teaching - supervision - juries

11.2.1 Teaching

- Master: Marcus Denker, 2 hours, Advanced Reflection. MetaLinks, VUB Brussels, Belgium.
- Master: Steven Costiou, Programmation orientée objet, Centrale Lille, 5h CM 5hTD
- Master: Steven Costiou, Live IoT avec Pharo, Polytech Lille, 6h CM 14hTP
- Master: Steven Costiou, Conception et modélisation objet, Polytech Lille, 12h CM 12hTD
- Master: Steven Costiou, Fondamentaux du debugging, Université de Lille, 8h CM, 8hTD
- Licence: Anne Etien, Programmation par objet, 50h, L3, Polytech Lille
- Master: Anne Etien, Test et Maintenance logicielle, 24h, M1, Polytech Lille
- Licence: Anne Etien, Introduction à la programmation, 40h, L1, Université de Lille
- Licence: Anne Etien, Technique du Web, 18, L2, Université de Lille
- Licence: Anne Etien, Introduction aux Bases de données, 36h, L3, Université de Lille
- Licence: Anne Etien, Bases de données relationnelles, 32h, L3, Université de Lille
- Licence: Vincent Aranega, Programmation Python, 56h, L1, Université de Lille, France
- Master: Vincent Aranega, Paradigme de Programmation par la Pratique, 80h, M1, Université de Lille, France
- Licence: Vincent Aranega, Pratique du C, 42h, L2, Université de Lille, France
- Licence: Vincent Aranega, Meta programmation Objet Avancé, 10h, L3, Université de Lille, France
- Licence: Vincent Aranega, Programmation avancée, 32h, L3, Polytech-Lille, France
- License: Damien Pollet, Introduction à la programmation, 40h, L1, Université de Lille
- License: Damien Pollet, Conception Orientée Objet, 18h, L3, Université de Lille
- License: Damien Pollet, Programmation des Systèmes, 21h, L3, Université de Lille
- License: Damien Pollet, Option Meta, 23h, L3, Université de Lille
- Master: Damien Pollet, Programmation Java, 40h, L3, Institut Mines-Telecom
- Master: Damien Pollet, Programmation Java avancée, 22h, L3, Institut Mines-Telecom
- Master: Damien Pollet, Techniques des Systèmes d'Information Ouverts, 17h, M1, Institut Mines-Telecom
- Master: Damien Pollet, Techniques des Systèmes d'Information Ouverts, 17h, M1, Institut Mines-Telecom
- Master: Damien Pollet, Ingénierie du Logiciel, 14h, M1, Institut Mines-Telecom
- Licence: Nicolas Anquetil, Principes des Système d'Exploitation, 37h, L2, IUT-A, Université de Lille, France
- Licence: Nicolas Anquetil, Conception OO Avancée, 48h, L2, IUT- A, Université de Lille, France
- Licence: Nicolas Anquetil, Modélisation Mathématique, 13h, L2, IUT-A, Université de Lille, France
- Licence: Nicolas Anquetil, Production d'Application, 30h, L2, IUT-A, Université de Lille, France

- Licence: Nicolas Anquetil, Programation Mobile, 30h, L2, IUT-A, Université de Lille, France
- Licence: Nicolas Anquetil, Projets Agiles, 12h, L2, IUT-A, Université de Lille, France
- Licence: Nicolas Anquetil, Interfaces Hommes-Machines, 32h, L2, IUT-A, Université de Lille, France
- Master: Santiago Bragagnolo, Navigation autonome , 28h, CM, M1, ISEN
- Master: Santiago Bragagnolo, Initiation aux technologies Blockchain, 6h, CM, M2, Universite de lille (MFCA)
- License: Santiago Bragagnolo, Tp Programmation Orientée Objet , 20h, Tp, M1 (Gis4), Polytech de Lille
- Master: Santiago Bragagnolo, Projet Programmation Orientée Objet , 24h, Tp, M1 (Gis4), Polytech de Lille
- License: Santiago Bragagnolo, Projet Programmation Orientée Objet , 24h, TP, L3 (Gis3), Polytech de Lille
- License: Thomas Dupriez, TD Algorithmique et Programmation, 18h, Td, L1, Université de Lille
- License: Thomas Dupriez, TP Algorithmique et Programmation, 18h, Tp, L1, Université de Lille
- License: Thomas Dupriez, Stage Unix, 12h, Tp, L2, Université de Lille

11.2.2 E-learning

- Pharo Mooc, 7 weeks, Licences and Master students

11.2.3 Supervision

- PhD: Julien Delplanque, *Software Engineering Techniques Applied to Databases*, September 2020, Anne Etien, Nicolas Anquetil
- PhD in progress: Oleksandr Zaitsev, *Machine Learning-Based Tools to Support Software Evolution*, started Jul 2019, Stéphane Ducasse, Nicolas Anquetil
- PhD in progress: Benoît Verhaeghe, *Support à l'automatisation de la migration d'interface d'applications Web : le cas de GWT vers Angular*, started Jan 2019, Anne Etien, Nicolas Anquetil
- PhD in progress: Théo Rogliano, *On multiple language kernel*, started Oct 2019, Stéphane Ducasse, Luc Fabresse
- PhD in progress: Pierre Misse-Chanabier, *Modular, green, versatile Virtual Machines*, started Oct 2019, Stéphane Ducasse, Noury Bouraqadi
- PhD in progress: Thomas Dupriez, *New Generation Debugger and Application Monitoring*, started Oct 2018, Stéphane Ducasse, Steven Costiou, Guillermo Polito
- PhD in progress: Mahugnon Honoré Houekpetodji, *Multi-Facet Actionable for Information System Rejuvenation*, SPI Lille, France, Stéphane Ducasse, Nicolas Anquetil, Nicolas Dias, Jérome Sudich
- PhD in progress: Carolina Hernández, *Tools for MicroKernels* Guillermo Polito and Luc Fabresse
- PhD in progress: Santiago Bragagnolo *Migration de programmes légataires vers des architectures Web: le cas de de la migration de programmes Microsoft Access vers Angular / Microservices*, CIFRE Berger-Levrault, Stéphane Ducasse, Nicolas Anquetil.
- PhD in progress: Maximilian Ignacio Willembrinck Santander, *Scriptable Time-Traveling Debuggers*, since october 2020, Inria, Anne Etien, Steven Costiou

11.2.4 Juries

- Anne Etien was the president of the PhD defense of José LOZANO APARICIO on 14th December 2020: *Échange de données de bases de données relationnelles vers RDF avec des schémas des contraintes sur cible.*

11.3 Popularization

11.3.1 Internal or external Inria responsibilities

- Inria Academy lectures (Stéphane Ducasse).
- Master class organised by Inria Chile "Advanced Design in Pharo" (Stéphane Ducasse).

11.3.2 Articles and contents

http://books.pharo.org contains two collections of books a technological one and a textbook one. Both collections are edited by S. Ducasse.

11.3.3 Education

A new version of the MOOC for Pharo was released (Stéphane Ducasse) for its fourth iteration. 70 new videos got produced. http://mooc.pharo.org

11.3.4 Interventions

- We have organized a FOSDEM 2020 participation for Pharo
  - Talk: A minimal pure object-oriented reflective language, Stéphane Ducasse / Pablo Tesone - FOSDEM 2020, Brussels, Belgium - 2/2/2020 https://archive.fosdem.org/2020/schedule/event/pharominimalreflectivelang/
  - Talk: Bootstrapping minimal reflective language kernels, Carolina Hernandez - FOSDEM 2020, Brussels, Belgium - 2/2/2020 https://archive.fosdem.org/2020/schedule/event/pharominimalrefllangkernels/
  - Talk: Pocket infrastructures to bridge reproducible research, live coding, civic hacktivism and data feminism for/from the Global South - FOSDEM 2020, Brussels, Belgium - 01/02/2020 https://archive.fosdem.org/2020/schedule/event/open_research_pocket_infrastructures/
- Multiple public remote Pharo Sprints in Lille. https://association.pharo.orgevents
- Talk: Vincent Aranega presentation for Brazil.
- Talk: JIT Compiler Infrastructure Talk in "Workshop Sistemas de Información", Pablo Tesone - Facultad Regional Delta - Universidad Tecnológica Nacional, Campana, Argentina. 15/07/2020 https://sistemaseventos.frd.utn.edu.ar/?page_id=27

12 Scientific production

12.1 Major publications

[1] S. Costiou, V. Aranega and M. Denker. 'Sub-method, partial behavioral reflection with Reflectivity: Looking back on 10 years of use'. In: *The Art, Science, and Engineering of Programming* 4.3 (Feb. 2020). DOI: 10.22152/programming-journal.org/2020/4/5. URL: https://hal.inria.fr/hal-02480136.

[2] J. Delplanque, A. Etien, N. Anquetil and S. Ducasse. 'Recommendations for Evolving Relational Databases'. In: *CAiSE 2020 - 32nd International Conference on Advanced Information Systems Engineering*. Grenoble, France, June 2020. URL: https://hal.inria.fr/hal-02511466.

[3] P. Tesone, S. Ducasse, G. Polito, L. Fabresse and N. Bouraqadi. 'A new modular implementation for Stateful Traits'. In: *Science of Computer Programming* 195 (Apr. 2020). DOI: 10.1016/j.scico.2020.102470. URL: https://hal.inria.fr/hal-02541842.

[4] P. Tesone, G. Polito, L. Fabresse, N. Bouraqadi and S. Ducasse. 'Preserving Instance State during Refactorings in Live Environments'. In: *Future Generation Computer Systems* (2020). DOI: 10.1016/j.future.2020.04.010. URL: https://hal.archives-ouvertes.fr/hal-02541754.

12.2 Publications of the year

International journals

[5] S. Costiou, V. Aranega and M. Denker. 'Sub-method, partial behavioral reflection with Reflectivity: Looking back on 10 years of use'. In: *The Art, Science, and Engineering of Programming* 4.3 (14th Feb. 2020). DOI: 10.22152/programming-journal.org/2020/4/5. URL: https://hal.inria.fr/hal-02480136.

[6] S. Costiou, M. Kerboeuf, C. Toullec, A. Plantec and S. Ducasse. 'Object Miners: Acquire, Capture and Replay Objects to Track Elusive Bugs.' In: *The Journal of Object Technology* 19.1 (2020), 1:1. DOI: 10.5381/jot.2020.19.1.a1. URL: https://hal.archives-ouvertes.fr/hal-02929746.

[7] G. A. Pierro, R. Tonelli and M. Marchesi. 'An Organized Repository of Ethereum Smart Contracts' Source Codes and Metrics'. In: *Future internet* 12.11 (Nov. 2020), p. 197. DOI: 10.3390/fi12110197. URL: https://hal.inria.fr/hal-03099061.

[8] P. Tesone, S. Ducasse, G. Polito, L. Fabresse and N. Bouraqadi. 'A new modular implementation for Stateful Traits'. In: *Science of Computer Programming* 195 (15th Apr. 2020). DOI: 10.1016/j.scico.2020.102470. URL: https://hal.inria.fr/hal-02541842.

[9] P. Tesone, G. Polito, L. Fabresse, N. Bouraqadi and S. Ducasse. 'Preserving Instance State during Refactorings in Live Environments'. In: *Future Generation Computer Systems* (2020). DOI: 10.1016/j.future.2020.04.010. URL: https://hal.archives-ouvertes.fr/hal-02541754.

International peer-reviewed conferences

[10] N. Anquetil, A. Etien, M. H. Houekpetodji, B. Verhaeghe, S. Ducasse, C. Toullec, F. Djareddir, J. Sudich and M. Derras. 'Modular Moose: A new generation software reverse engineering environment'. In: International Conference on Software Reuse. Tunis, Tunisia, 6th Oct. 2020. URL: https://hal.inria.fr/hal-02972159.

[11] S. Bragagnolo, N. Anquetil, S. Ducasse, A. Seriai and M. Derras. 'Analysing Microsoft Access Projects: Building a model in a Partially Observable Domain'. In: ICSR 2020. Hammamet, Tunisia, 2nd Dec. 2020. URL: https://hal.inria.fr/hal-02966146.

[12] S. Bragagnolo, B. Verhaeghe, A. Seriai, M. Derras and A. Etien. 'Challenges for Layout Validation: Lessons Learned'. In: QUATIC 2020 - 13th International Conference on the Quality of Information and Communications Technology. Faro, Portugal, 8th Sept. 2020. URL: https://hal.inria.fr/hal-02914750.

[13] J. Delplanque, A. Etien, N. Anquetil and S. Ducasse. 'Recommendations for Evolving Relational Databases'. In: CAiSE 2020 - 32nd International Conference on Advanced Information Systems Engineering. Grenoble, France, 8th June 2020. URL: https://hal.inria.fr/hal-02511466.

[14] S. Ducasse, L. Dargaud and G. Polito. 'Microdown: a clean and extensible markup language to support Pharo documentation'. In: International Workshop of Smalltalk Technologies. virtual, France, 4th Nov. 2020. URL: https://hal.inria.fr/hal-03137098.

[15] H. Gabsi, R. Drira, H. Hajjami Ben Ghézala and S. Ducasse. 'From Business Process to Cloud Application'. In: IBIMA 2020 - International Business Information Management Association Conference. Seville, Spain, 1st Apr. 2020. URL: https://hal.inria.fr/hal-02533375.

[16] P. Laborde, S. Costiou, É. Le Pors and A. Plantec. '15 years of reuse experience in evolutionary prototyping for the defense industry'. In: International Conference on Software and Systems Reuse. Hammamet, Tunisia, 2nd Dec. 2020. URL: https://hal.inria.fr/hal-02966691.

[17] M. Marra, G. Polito and E. Gonzalez Boix. 'Framework-aware debugging with stack tailoring'. In: SPLASH '20: Conference on Systems, Programming, Languages, and Applications, Software for Humanity. Virtual USA, United States, 15th Nov. 2020, pp. 71–84. DOI: 10.1145/3426422.3426982. URL: https://hal.inria.fr/hal-03043779.

[18] G. A. Pierro, A. Bergel, R. Tonelli and S. Ducasse. 'An Interdisciplinary Model for Graphical Representation'. In: CIFMA 2020 - 2nd International Workshop on Cognition: Interdisciplinary Foundations, Models and Applications. Amsterdam / Virtual, Netherlands, 1st Oct. 2020. URL: https://hal.inria.fr/hal-02972186.

[19] O. Zaitsev, S. Ducasse, A. Bergel and M. Eveillard. 'Suggesting Descriptive Method Names: An Exploratory Study of Two Machine Learning Approaches'. In: QUATIC 2020 - 13th International Conference on the Quality of Information and Communications Technology. Faro / Virtual, Portugal: https://2020.quatic.org/, 8th Sept. 2020. URL: https://hal.inria.fr/hal-02962334.

Scientific books

[20] O. Auverlot, S. Ducasse and L. Fabresse. *TinyBlog: Develop your First Web App with Pharo*. 1st Apr. 2020. URL: https://hal.archives-ouvertes.fr/hal-02297688.

Reports & preprints

[21] S. Costiou, T. Dupriez and S. Ducasse. *New Generation Debuggers: Défis du GDRGPL 2020*. Inria Lille - Nord Europe; GDR GPL, 11th Nov. 2020. URL: https://hal.inria.fr/hal-02999965.

[22] S. Costiou, T. Dupriez and D. Pollet. *Handling Error-Handling Errors: dealing with debugger bugs in Pharo: Preprint from IWST20: International Workshop on Smalltalk Technologies*. 18th Nov. 2020. URL: https://hal.inria.fr/hal-02992644.

[23] J. Delplanque, A. Etien, N. Anquetil and S. Ducasse. *Recommendations for Evolving Relational Databases: Technical Report*. Univ. Lille, CNRS, Centrale Lille, Inria UMR 9189 - CRIStAL,INRIA Lille Nord Europe, Villeneuve d'Ascq, France, 11th Mar. 2020. URL: https://hal.inria.fr/hal-02504949.

[24] M. Denker, N. Anquetil, V. Aranega, S. Costiou, S. Ducasse, A. Etien and D. Pollet. *Project-Team RMoD 2019 Activity Report*. INRIA, 1st Mar. 2020. URL: https://hal.inria.fr/hal-03120443.

[25] T. Dupriez, S. Costiou and S. Ducasse. *First Infrastructure and Experimentation in Echo-debugging: Preprint from IWST20: International Workshop on Smalltalk Technologies*. 18th Nov. 2020. URL: https://hal.inria.fr/hal-02992863.

[26] P. Laborde, S. Costiou, A. Plantec and E. Le Pors. *Molecule: live prototyping with component-oriented programming: Preprint from IWST20: International Workshop on Smalltalk Technologies*. Novi Sad, Serbia, 18th Nov. 2020. URL: https://hal.inria.fr/hal-02966704.

[27] R. Salgado, M. Denker, S. Ducasse, A. Etien and V. Aranega. *Towards a Smart Data Processing and Storage Model: Preprint from IWST20: International Workshop on Smalltalk Technologies*. Novi Sad, Serbia, 18th Nov. 2020. URL: https://hal.archives-ouvertes.fr/hal-03101646.

[28] O. Zaitsev, S. Ducasse and N. Anquetil. *Characterizing Pharo Code: A Technical Report*. Inria Lille Nord Europe - Laboratoire CRIStAL - Université de Lille; Arolla, 14th Jan. 2020. URL: https://hal.inria.fr/hal-02440055.

Other scientific publications

[29] N. Margulies. 'Méta-interprétation pour la génération de compilateur Just-In-Time'. University Lille 1, 1st Sept. 2020. URL: https://hal.inria.fr/hal-03012007.

12.3 Cited publications

[30] N. Anquetil. 'A Comparison of Graphs of Concept for Reverse Engineering'. In: *Proceedings of the 8th International Workshop on Program Comprehension*. IWPC'00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 231–240. URL: http://rmod.lille.inria.fr/archives/papers/Anqu00b-ICSM-GraphsConcepts.pdf.

[31] A. Bergel, S. Ducasse and 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. URL: http://scg.unibe.ch/archive/papers/Berg05bclassboxjOOPSLA.pdf.

[32] A. Bergel, S. Ducasse, O. Nierstrasz and R. Wuyts. 'Stateful Traits'. In: *Advances in Smalltalk — Proceedings of 14th International Smalltalk Conference (ISC 2006)*. Vol. 4406. LNCS. Springer, Aug. 2007, pp. 66–90. DOI: 10.1007/978-3-540-71836-9_3. URL: http://dx.doi.org/10.1007/978-3-540-71836-9%5C_3.

[33] A. Bergel, S. Ducasse, O. Nierstrasz and R. Wuyts. 'Stateful Traits and their Formalization'. In: *Journal of Computer Languages, Systems and Structures* 34.2-3 (2008), pp. 83–108. DOI: 10.1016/j.cl.2007.05.003. URL: http://dx.doi.org/10.1016/j.cl.2007.05.003.

[34] A. P. Black, N. Schärli and 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)*. Vol. 38. Oct. 2003, pp. 47–64. DOI: 10.1145/949305.949311. URL: http://scg.unibe.ch/archive/papers/Blac03aTraitsHierarchy.pdf.

[35] G. Bracha and 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. URL: http://bracha.org/mirrors.pdf.

[36] D. Caromel and J. Vayssière. 'A security framework for reflective Java applications'. In: *Software: Practice and Experience* 33.9 (2003), pp. 821–846. DOI: 10.1002/spe.528. URL: http://dx.doi.org/10.1002/spe.528.

[37] D. Caromel and 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.

[38] P. Cointe. 'Metaclasses are First Class: the ObjVlisp Model'. In: *Proceedings OOPSLA '87, ACM SIGPLAN Notices*. Vol. 22. Dec. 1987, pp. 156–167.

[39] S. Denier. 'Traits Programming with AspectJ'. In: *Actes de la Première Journée Francophone sur le Développement du Logiciel par Aspects (JFDLPA'04)*. Ed. by P. Cointe. Paris, France, Sept. 2004, pp. 62–78.

[40] S. Ducasse and T. Gîrba. 'Using Smalltalk as a Reflective Executable Meta-Language'. In: *International Conference on Model Driven Engineering Languages and Systems (Models/UML 2006)*. Vol. 4199. LNCS. Berlin, Germany: Springer-Verlag, 2006, pp. 604–618. DOI: 10.1007/11880240_42. URL: http://scg.unibe.ch/archive/papers/Duca06dMOOSEMODELS2006.pdf.

[41] S. Ducasse, T. Gîrba, M. Lanza and S. Demeyer. 'Moose: a Collaborative and Extensible Reengineering Environment'. In: *Tools for Software Maintenance and Reengineering*. RCOST / Software Technology Series. Milano: Franco Angeli, 2005, pp. 55–71. URL: http://scg.unibe.ch/archive/papers/Duca05aMooseBookChapter.pdf.

[42] S. Ducasse, O. Nierstrasz, N. Schärli, R. Wuyts and A. P. Black. 'Traits: A Mechanism for fine-grained Reuse'. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 28.2 (Mar. 2006), pp. 331–388. DOI: 10.1145/1119479.1119483. URL: http://scg.unibe.ch/archive/papers/Duca06bTOPLASTraits.pdf.

[43] S. Ducasse, R. Wuyts, A. Bergel and 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)*. Montreal, Quebec, Canada: ACM Press, Oct. 2007, pp. 171–190. DOI: 10.1145/1297027.1297040. URL: http://scg.unibe.ch/archive/papers/Duca07b-FreezableTrait.pdf.

[44] A. Dunsmore, M. Roper and M. Wood. 'Object-Oriented Inspection in the Face of Delocalisation'. In: *Proceedings of ICSE '00 (22nd International Conference on Software Engineering)*. Limerick, Ireland: ACM Press, 2000, pp. 467–476.

[45] K. Fisher and J. Reppy. *Statically typed traits*. Technical Report TR-2003-13. University of Chicago, Department of Computer Science, Dec. 2003.

[46] P. W. L. Fong and C. Zhang. *Capabilities as alias control: Secure cooperation in dynamically extensible systems*. Tech. rep. Department of Computer Science, University of Regina, 2004.

[47] M. Furr, J.-h. An and J. S. Foster. 'Profile-guided static typing for dynamic scripting languages'. In: *OOPSLA'09*. 2009.

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

[49] L. Gong. 'New security architectural directions for Java'. In: *compcon* 0 (1997), p. 97. DOI: 10.1109/CMPCON.1997.584679. URL: http://dx.doi.org/10.1109/CMPCON.1997.584679.

[50] M. Hicks and S. Nettles. 'Dynamic software updating'. In: *ACM Transactions on Programming Languages and Systems* 27.6 (Nov. 2005), pp. 1049–1096. DOI: 10.1145/1108970.1108971. URL: http://dx.doi.org/10.1145/1108970.1108971.

[51] G. Kiczales, J. des Rivières and D. G. Bobrow. *The Art of the Metaobject Protocol*. MIT Press, 1991.

[52] G. Kiczales and L. Rodriguez. 'Efficient Method Dispatch in PCL'. In: *Proceedings of ACM conference on Lisp and Functional Programming*. Nice, 1990, pp. 99–105.

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

[54] S. Liang and G. Bracha. 'Dynamic Class Loading in the Java Virtual Machine'. In: *Proceedings of OOPSLA '98, ACM SIGPLAN Notices*. 1998, pp. 36–44.

[55] L. Liquori and A. Spiwack. 'FeatherTrait: A Modest Extension of Featherweight Java'. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 30.2 (2008), pp. 1–32. DOI: 10.1145/1330017.1330022. URL: http://www-sop.inria.fr/members/Luigi.Liquori/PAPERS/toplas-07.pdf.

[56] B. Livshits and T. Zimmermann. 'DynaMine: finding common error patterns by mining software revision histories'. In: *SIGSOFT Software Engineering Notes* 30.5 (Sept. 2005), pp. 296–305.

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

[58] M. S. Miller. 'Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control'. PhD thesis. Baltimore, Maryland, USA: Johns Hopkins University, May 2006.

[59] M. S. Miller, C. Morningstar and B. Frantz. 'Capability-based Financial Instruments'. In: *FC '00: Proceedings of the 4th International Conference on Financial Cryptography*. Vol. 1962. Springer-Verlag, 2001, pp. 349–378.

[60] O. Nierstrasz, S. Ducasse and N. Schärli. 'Flattening Traits'. In: *Journal of Object Technology* 5.4 (May 2006), pp. 129–148. URL: http://www.jot.fm/issues/issue_2006_05/article4.

[61] P. J. Quitslund. *Java Traits — Improving Opportunities for Reuse*. Technical Report CSE-04-005. Beaverton, Oregon, USA: OGI School of Science & Engineering, Sept. 2004.

[62] J. Reppy and A. Turon. 'A Foundation for Trait-based Metaprogramming'. In: *International Workshop on Foundations and Developments of Object-Oriented Languages*. 2006.

[63] F. Rivard. 'Pour un lien d'instanciation dynamique dans les langages à classes'. In: *JFLA96*. INRIA — collection didactique, Jan. 1996.

[64] J. H. Saltzer and M. D. Schoroeder. 'The Protection of Information in Computer Systems'. In: *Fourth ACM Symposium on Operating System Principles*. Vol. 63. IEEE, Sept. 1975, pp. 1278–1308.

[65] N. Sangal, E. Jordan, V. Sinha and D. Jackson. 'Using Dependency Models to Manage Complex Software Architecture'. In: *Proceedings of OOPSLA'05*. 2005, pp. 167–176.

[66] N. Schärli, A. P. Black and 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)*. Oct. 2004, pp. 130–149. DOI: 10.1145/1028976.1028988. URL: http://scg.unibe.ch/archive/papers/Scha04b00Encapsulation.pdf.

[67] N. Schärli, S. Ducasse, O. Nierstrasz and A. P. Black. 'Traits: Composable Units of Behavior'. In: *Proceedings of European Conference on Object-Oriented Programming (ECOOP'03)*. Vol. 2743. LNCS. Springer Verlag, July 2003, pp. 248–274. DOI: 10.1007/b11832. URL: http://scg.unibe.ch/archive/papers/Scha03aTraits.pdf.

[68] C. Smith and S. Drossopoulou. 'Chai: Typed Traits in Java'. In: *Proceedings ECOOP 2005*. 2005.

[69] G. Snelting and F. Tip. 'Reengineering Class Hierarchies using Concept Analysis'. In: *ACM Trans. Programming Languages and Systems*. 1998.

[70] K. J. Sullivan, W. G. Griswold, Y. Cai and B. Hallen. 'The Structure and Value of Modularity in SOftware Design'. In: *ESEC/FSE 2001*. 2001.

[71] D. Vainsencher. 'MudPie: layers in the ball of mud'. In: *Computer Languages, Systems & Structures* 30.1-2 (2004), pp. 5–19.

[72] N. Wilde and R. Huitt. 'Maintenance Support for Object-Oriented Programs'. In: *IEEE Transactions on Software Engineering* SE-18.12 (Dec. 1992), pp. 1038–1044.