IN PARTNERSHIP WITH:
Université de Lille

Activity Report 2019

Project-Team RMOD

Analyses and Languages Constructs for Object-Oriented Application Evolution

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

RESEARCH CENTER
Lille - Nord Europe

THEME
Distributed programming and Software engineering

Table of contents

Project-Team RMOD

Creation of the Project-Team: 2009 July 01

Keywords:

Computer Science and Digital Science:

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, Visitors, External Collaborators

Research Scientists

Stéphane Ducasse [Team leader, Inria, Senior Researcher, HDR]
Steven Costiou [Inria, Researcher, from Oct 2019]
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, Associate Professor, HDR]
Damien Pollet [Université de Lille, Associate Professor]

Post-Doctoral Fellow

Steven Costiou [Inria, Post-Doctoral Fellow, until Sep 2019]

PhD Students

Lionel Akue [Inria, PhD Student, until Apr 2019]
Julien Delplanque [Université de Lille, PhD Student]

Thomas Dupriez [Université de Lille, PhD Student]
Carolina Hernández Phillips [Inria, PhD Student]
Mahugnon Honoré Houekpetodji [Inria, PhD Student]
Jason Lecerf [CEA, PhD Student, until Nov 2019]
Matteo Marra [VUB Brussels, Belgium, PhD Student]
Pierre Misse Chanabier [Inria, PhD Student, from Oct 2019]
Théo Rogliano [Inria, PhD Student, from Oct 2019]
Pablo Tesone [École des Mines de Douai, PhD Student, until Mar 2019]
Benoît Verhaeghe [Berger-Levrault, PhD Student]
Oleksandr Zaitsev [Inria, PhD Student, from Mar 2019]

Technical staff

Andy Amoordon [Inria, Engineer, from Apr 2019 until Jul 2019]
Santiago Bragagnolo [Inria, Engineer, from Apr 2019]
Cyril Ferlicot-Delbecque [Inria, Engineer]
Guillaume Larchevêque [Inria, Engineer, until Oct 2019]
Allex Oliveira [Inria, Engineer]
Pablo Tesone [Inria, Engineer, from Apr 2019]
Christophe Demarey [Inria, Engineer, 60%]

Interns and Apprentices

Vanille Decoeyere [Inria, from Dec 2019]
Clément Dutriez [Université de Lille, from Apr 2019 until Aug 2019]
Dayne Lorena Guerra Calle [Inria, from Feb 2019 until Aug 2019]
Hugo Lasnier [Université de Lille, from Apr 2019 until Aug 2019]
Chia Yu Li [Inria, until Jul 2019]
Pierre Misse Chanabier [Inria, from Feb 2019 until Jul 2019]
Théo Rogliano [Inria, from Feb 2019 until Jul 2019]
Iona Thomas [Centrale Lille, from Jul 2019 until Aug 2019]
Clotilde Toullec [Université de Lille, from May 2019 until Aug 2019]
Oleksandr Zaitsev [Inria, until Feb 2019]

Administrative Assistant

Julie Jonas [Inria]

Visiting Scientists

Abdelhakim Bouremel [University of Skikda, Algeria, Oct 2019]
Mohamad Chakroun [Independent, Mar 2019]
Victor Martín Dias [University of Chile, Chile, until Sep 2019]
Christopher Fuhrman [École de technologie supérieure de Montréal, Canada, until Sep 2019]
Yann-Gaël Guéhéneuc [Concordia University, Canada, from Apr 2019 until May 2019]
Sebastijan Kaplar [University of Novi Sad, Serbia, Aug 2019]
Manuel Leuenberger [University of Bern, Switzerland, from Sep 2019 until Nov 2019]
Milton Mamani Torres [Object Profile SpA, Chile, Aug 2019]
Nina Medic [University of Novi Sad, Serbia, from Jun 2019 until Jul 2019]
Hayatou Oumarou [University of Maroua, Cameroun from Sep 2019 until Oct 2019]
Giuseppe Antonio Pierro [University of Cagliari, Italy, until July 2019]
Gordana Rakic [University of Novi Sad, Serbia, from Nov 2019]
Moussa Saker [University Badji Mokhtar-Annaba, Algeria, from Nov 2019]

External Collaborators

Luc Fabresse [IMT Lille-Douai]
Pavel Krivanek [Inria, until Jun 2019]
Guillermo Polito [CNRS]
Esteban Lorenzano [InriaSoft]

Olivier Auverlot [Université de Lille]

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 [40]. 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 [37]. 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 [46]. In reflective languages such as F-Script, Ruby, Python, Lua, JavaScript, Smalltalk and Java (to a certain extent), it is possible to nearly change any aspect of an application: change objects, change classes dynamically, migrate instances, and even load untrusted code.

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

What are the language modularity constructs to support isolation?

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

3. Research Program

3.1. Software Reengineering

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

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

To help us answer these questions, we work on enriching Moose, our reengineering environment, with a new set of analyses [31], [30]. 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) [59] 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 [20]. We look for solutions to help people putting FCA to real use.

[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.

3.1.2. Remodularization analyses

Context and Problems. It is a well-known practice to layer applications with bottom layers being more stable than top layers [47]. Until now, few works have attempted to identify layers in practice: Mudpie [61] is a first cut at identifying cycles between packages as well as package groups potentially representing layers. DSM (dependency structure matrix) [60], [55] 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 [43]. 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 [62], [34].

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 [57], [32] and classboxes [21] 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 [57], [32]. 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 [29], [51], [22], [33] and several type systems were defined [35], [58], [52], [45].

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 [24]. 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 [32], stateful [23], and freezable [33]) 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 [50]. 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 [28] then from Smalltalk [38].

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 [42]). 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 [25], as well as controlling the access to reflective features [26], [27] are important challenges. We plan to:

- Study the isolation abstractions available in erights (http://www.erights.org) [49], [48], and Java's class loader strategies [44], [39].
- Categorize the different reflective features of languages such as CLOS [41], Python and Smalltalk [53] and identify suitable isolation mechanisms and infrastructure [36].
- Assess different isolation models (access rights, capabilities [54],...) 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 [25],
  - the use of encapsulation policies as a basis to restrict the interfaces of the controlled objects [56],
  - 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,...) [53].

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. Highlights of the Year

5.1. Highlights of the Year

- Steven Costiou was hired as CR.
- We released Pharo 7. More information at http://pharo.org.
- The paper *Rotten Green Tests* has been accepted at ICSE. https://hal.inria.fr/hal-02002346

5.1.1. Awards

- Best paper award: *SATToSE 2019 Migrating GWT to angular 6 using MDE*.
  https://hal.inria.fr/hal-02304301
- 2nd place best paper award IWST 2019: *Illicium: a modular transpilation toolchain from Pharo to C*.
  https://hal.archives-ouvertes.fr/hal-02297860
- 3rd place best paper award IWST 2019: *GildaVM: a Non-Blocking I/O Architecture for the Cog VM*.
  https://hal.archives-ouvertes.fr/view/index/docid/2379275

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, Stéphane Ducasse, Julien Delplanque, Guillaume Larcheveque, Cyril Ferlicot-Delbecque and Pavel Krivanek
- 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.

- Participants: Christophe Demarey, Clement Bera, Damien Pollet, Esteban Lorenzano, Marcus Denker, Stéphane Ducasse and Guillermo Polito
- Partners: 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
- 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. It is used in the tools developed by Feenk.com.

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

7. New Results

7.1. Dynamic Languages: Virtual Machines

Illicium A modular transpilation toolchain from Pharo to C. The Pharo programming language runs on the OpenSmalltalk-VM. This Virtual Machine (VM) is mainly written in Slang, a subset of the Smalltalk language dedicated to VM development. Slang is transpiled to C using the Slang-to-C transpiler. The generated C is then compiled to produce the VM executable binary code. Slang is a powerful dialect for generating C because it benefits from the tools of the Smalltalk environment, including a simulator that runs and debugs the VM. However, the Slang-to-C transpiler is often too permissive. For example, the Slang-to-C transpiler generates invalid C code from some Smalltalk concepts it does not support. This makes the Slang code hard to debug as the errors are caught very late during the development process, which is worsen by the loss of the mapping between the generated C code and Slang. The Slang-to-C transpiler is also hard to extend or adapt to modify part of the translation process. We present Illicium, a new modular transpilation toolchain based on a subset of Pharo targeting C through AST transformations. This toolchain translates the Pharo AST into a C AST to generate C code. Using ASTs as source and target artifacts enables analysis, modification and validation at different levels during the translation process. The main translator is split into smaller and replaceable translators to increase modularity. Illicium also allows the possibility to introduce new translators and to chain them together, increasing reusability. To evaluate our approach, we show with a use case how to extend the transpilation process with a translation that requires changes not considered in the original C AST. [7]

GildaVM: a Non-Blocking I/O Architecture for the Cog VM. The OpenSmalltalk virtual machine (VM) was historically designed as a single-threaded VM. All VM code including the Smalltalk interpreter, the garbage collector and the just-in-time compiler run in the same single native thread. While this VM provides concurrency through green threads, it cannot take advantage of multi-core processors. This architecture performs really well in practice until the VM accesses external resources such as e.g., FFI callouts, which block the single VM thread and prevent green threads to benefit from the processor. We present GildaVM, a multi-threaded VM architecture where one thread at a time executes the VM while allowing non-blocking I/O in parallel. The ownership of the VM is orchestrated by a Global Interpreter Lock (GIL) as in the standard implementations of Python and Ruby. However, within a single VM thread concurrency is still possible through green threads. We present a prototype implementation of this architecture running on top of the Stack flavour of the OpenSmalltalk VM. We finally evaluate several aspects of this architecture like FFI and thread-switch overhead. While current benchmarks show good results for long FFI calls, short FFI calls require more research to minimize the overhead of thread-switch. [9]

7.2. Dynamic Languages: Language Constructs for Modular Design

Magic Literals in Pharo Literals are constant values (numbers, strings, etc.) used in the source code. Magic literals are the ones used without a clear explanation of their meaning. Presence of such literals harms source code readability, decreases its modularity, and encourages code duplication. Identifying magic literals is not straightforward. A literal can be considered self-explanatory in one context and magic in another. We

need a heuristic to help developers spot magic literals. We study and characterize the literals in Pharo. We implemented a heuristic to detect magic literals and integrated it as a code critic rule for System Browser and Critics Browser in Pharo 7. We run our heuristic on 112,500 Pharo methods which reported 23,292 magic literals spread across 8,986 methods. We manually validated our approach on a random subset of 100 methods and found that 62% of the reported literals in those methods are indeed magic. [3]

Towards easy program migration using language virtualization Migrating programs between language versions is a daunting task. A developer writes a program in a particular version of a language and cannot foresee future language changes. In this article, we explore a solution to gradual program migration based on virtualization at the programming language level. Our language virtualization approach adds a backwards-compatibility layer on top of a recent language version, allowing developers to load and run old programs on the more recent infrastructure. Developers are then able to migrate the program to the new language version or are able to run it as it is. Our virtualization technique is based on a dynamic module implementation and code intercession techniques. Migrated and non-migrated parts co-exist in the meantime allowing an incremental migration procedure. We validate it by migrating legacy Pharo programs, MuTalk and Fuel. [10]

7.3. Dynamic Languages: Debugging

Sindarin: A Versatile Scripting API for the Pharo Debugger Debugging is one of the most important and time consuming activities in software maintenance, yet mainstream debuggers are not well-adapted to several debugging scenarios. This has led to the research of new techniques covering specific families of complex bugs. Notably, recent research proposes to empower developers with scripting DSLs, plugin-based and moldable debuggers. However, these solutions are tailored to specific use-cases, or too costly for one-time-use scenarios. We argue that exposing a debugging scripting interface in mainstream debuggers helps in solving many challenging debugging scenarios. For this purpose, we present Sindarin, a scripting API that eases the expression and automation of different strategies developers pursue during their debugging sessions. Sindarin provides a GDB-like API, augmented with AST-bytecode-source code mappings and object-centric capabilities. To demonstrate the versatility of Sindarin, we reproduce several advanced breakpoints and non-trivial debugging mechanisms from the literature. [4]

Challenges in Debugging Bootstraps of Reflective Kernels The current explosion of embedded systems (i.e., IoT, Edge Computing) implies the need for generating tailored and customized software for these systems. Instead of using specific runtimes (e.g., MicroPython, eLua, mRuby), we advocate that bootstrapping specific language kernels is a promising higher-level approach because the process takes advantage of the generated language abstractions, easing the task for a language developer. Nevertheless, bootstrapping language kernels is still challenging because current debugging tools are not suitable for fixing the possible failures that occur during the process. We take the Pharo bootstrap process as an example to analyse the different challenges a language developer faces. We propose a taxonomy of failures appearing during bootstrap and their causes. Based on this analysis, we identify future research directions: (1) prevention measures based on the reification of implicit virtual machine contracts, and (2) hybrid debugging tools that unify the debugging of high-level code from the bootstrapped language with low-level code from the virtual machine. [6]

7.4. Software Reengineering

Decomposing God Classes at Siemens A group of developers at Siemens Digital Industry Division approached our team to help them restructure a large legacy system. Several problems were identified, including the presence of God classes (big classes with thousands of lines of code and hundred of methods). They had tried different approaches considering the dependencies between the classes, but none were satisfactory. Through interaction during the last three years with a lead software architect of the project, we designed a software visualization tool and an accompanying process that allows her to propose a decomposition of a God Class in a matter of one or two hours even without prior knowledge of the class (although actually implementing the decomposition in the source code could take a week of work). We present the process that was formalized to decompose God Classes and the tool that was designed. We give details on the system itself and some of the classes that were decomposed. The presented process and visualisations have been successfully used for the last three years on a real industrial system at Siemens. [1]

Rotten Green Tests Unit tests are a tenant of agile programming methodologies, and are widely used to improve code quality and prevent code regression. A green (passing) test is usually taken as a robust sign that the code under test is valid. However, some green tests contain assertions that are never executed. We call such tests Rotten Green Tests. Rotten Green Tests represent a case worse than a broken test: they report that the code under test is valid, but in fact do not test that validity. We describe an approach to identify rotten green tests by combining simple static and dynamic call-site analyses. Our approach takes into account test helper methods, inherited helpers, and trait compositions, and has been implemented in a tool called DrTest. DrTest reports no false negatives, yet it still reports some false positives due to conditional use or multiple test contexts. Using DrTest we conducted an empirical evaluation of 19,905 real test cases in mature projects of the Pharo ecosystem. The results of the evaluation show that the tool is effective; it detected 294 tests as rotten-green tests that contain assertions that are not executed. Some rotten tests have been "sleeping" in Pharo for at least 5 years. [2]

Migrating GWT to Angular 6 using MDE In the context of a collaboration with Berger-Levrault, a major IT company, we are working on the migration of a GWT application to Angular. We focus on the GUI aspect of this migration which, even if both are web frameworks, is made difficult because they use different programming languages (Java for one, Typescript for the other) and different organization schemas (e.g. different XML files). Moreover, the new application must mimic closely the visual aspect of the old one so that the users of the application are not disturbed. We propose an approach in three steps that uses a meta-model to represent the GUI at a high abstraction level. We evaluated this approach on an application comprising 470 Java (GWT) classes representing 56 screens. We are able to model all the web pages of the application and 93% of the wid-gets they contain, and we successfully migrated (i.e., the result is visually equal to the original) 26 out of 39 pages (66%). We give examples of the migrated pages, both successful and not. [14] [11] [12]

Empirical Study of Programming to an Interface A popular recommendation to programmers in object-oriented software is to *program to an interface, not an implementation* (PTI). Expected benefits include increased simplicity from abstraction, decreased dependency on implementations, and higher flexibility. Yet, interfaces must be immutable, excessive class hierarchies can be a form of complexity, and *speculative generality* is a known code smell. To advance the empirical knowledge of PTI, we conducted an empirical investigation that involves 126 Java projects on GitHub, aiming to measuring the decreased dependency benefits (in terms of cochange). [13]

Exposing Test Analysis Results with DrTests Tests are getting the cornerstone of continuous development process and software evolution. Tests are the new gold. To improve test quality, a plethora of analyses is proposed such as test smells, mutation testing, test coverage. The problem is that each analysis often needs a particular way to expose its results to the developer. There is a need for an architecture supporting test running and analysis in a modular and extensible way. We present an extensible plugin-based architecture to run and report test results. DrTests is a new test browser that implements such plugin-based architecture. DrTests supports the execution of rotten tests, comments to tests, coverage and profiling tests. [5]

7.5. Blockchain Software Engineering

SmartAnvil: Open-Source Tool Suite for Smart Contract Analysis Smart contracts are new computational units with special properties: they act as classes with aspectual concerns; their memory structure is more complex than mere objects; they are obscure in the sense that once deployed it is difficult to access their internal state; they reside in an append-only chain. There is a need to support the building of new generation tools to help developers. Such support should tackle several important aspects: (1) the static structure of the contract, (2) the object nature of published contracts, and (3) the overall data chain composed of blocks and transactions. We present SmartAnvil an open platform to build software analysis tools around smart contracts. We illustrate the general components and we focus on three important aspects: support for static analysis of Solidity smart contracts, deployed smart contract binary analysis through inspection, and blockchain navigation and querying. SmartAnvil is open-source and supports a bridge to the Moose data and software analysis platform. [18]

The Influence Factors on Ethereum Transaction Fees In Ethereum blockchain, the user needs to set a Gas price to get a transaction processed and approved by Miners. To have the transaction executed, the Gas price has to be greater than or equal to the lowest Ethereum transaction fees. We present a set of data sampled every 15 seconds, from December 1st, 2018 to December 15, 2018, coming from different blockchain web APIs. The aim is to investigate whether and to what extent different variables - such as the number of pending transactions, the value of the USD/Ether pair, average electricity prices around the world, and the number of miners - influence the Ethereum transaction fees. This study is relevant from an economic perspective because more and more companies in different economic fields are adopting Ethereum blockchain. From historical data analysis, we found that only some of these variables do have an influence. For example, the number of pending transactions and the number of miners have a major influence on Ethereum transaction fees when compared to the other variables. [8]

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

8.1.1. 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.

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

8.2. Bilateral Grants with Industry

8.2.1. 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 à l'automatisation de la migration d'interface d'applications Web : le cas de GWT vers Angular*.

8.2.2. Arolla: Machine Learning-Based Recommenders to Support Software Evolution

Participants: Nicolas Anquetil, Stéphane Ducasse, Anne Etien, Oleksandr Zaitsev

We started a new collaboration with the council company, Arolla, about software evolution. Arolla has daily problems with identifying architecture, design, and deviations from those artifacts. The goal of the thesis is to experiment which learning techniques can help with semi-automatically extracting design and architectural aspects and their violations. A PhD CIFRE has started in 2019: Oleksandr Zaitsev, *Machine Learning-Based Tools to Support Software Evolution*.

A second CIFRE around legacy system cartography will start in 2020.

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.

Pablo Tesone did a PhD co-supervided by RMOD and Pr. L. Fabresse and N. Bouraqadi (finished in 2018). Currently, three PhD Students are co-supervised:

- 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: Carolina Hernández, *Tools for MicroKernels* Guillermo Polito and Luc Fabresse

We are collaborating in the Context of CPER Data since 2018.

9.1.2. CPER DATA

Participants: Marcus Denker, Stéphane Ducasse, Allex Oliveira with: L. Fabresse and N. Bouraqadi (IMT Douai)
From 2018, ongoing.

Funding to work one year on the PharoThings Platform. We are creating content for a website and a Demo in callaboration with IMT Douai.

9.2. National Initiatives

9.2.1. CEA List

Participants: Jason Lecerf, Stéphane Ducasse with Thierry Goubier (CEA List)
From 2016, PhD finished 2019.

Jason Lecerf started a shared PhD Oct 2016 and finished November 2019: *Designing Language-Agnostic Code Transformation Engines*.

9.3. European Initiatives

9.3.1. Collaborations in European Programs, Except FP7 & H2020

University of Novi Sad, Serbia
Participants: Stéphane Ducasse, Anne Etien, Nicolas Anquetil, Vincent Aranega

A collaboration with the University of Novi Sad, Serbia, started in 2018 with the university joining the Pharo Consortium as an academic member.

We have handed in a bilateral project (Campus France) between Novi Sad and RMOD: *An innovative visual environment in service of developer experience*. We expect results by the end of this year.

A Master thesis has been cosupervised. Nina Medic: *Graph library with layout algorithms in Pharo*.

Visitors:

- Sebastijan Kaplar [University of Novi Sad, Serbia, Aug 2019]
- Gordana Rakic [University of Novi Sad, Serbia, from Nov 2019]
- Nina Medic [University of Novi Sad, Serbia, from Jun 2019 until Jul 2019]

University of Prague, Czech Republic
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 2020.

University of Cagliari, Italy
Participants: Stéphane Ducasse

We are working on sofware engineering problems in the context of blockchain based sofware.

Visitor: Giuseppe Antonio Pierro [University of Cagliari, until July 2019].

University of Bern, Switzerland
Participants: Stéphane Ducasse, Marcus Denker

We are working on dynamic software update to, for example, automatically transform users of deprecated code.

Visitor: Manuel Leuenberger [University of Bern, from Sep 2019 until Nov 2019]

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* [1].

9.4. International Initiatives

9.4.1. Inria International Labs

Discussions with Inria Chile have started about organizing Pharo lectures in Chile. A first visit to Inria Chile in fall 2019 did not happen due to the political situation in Chile.

9.4.2. Inria Associate Teams Not Involved in an Inria International Labs

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

Collaboration with SOFT started 2016, from 2020 Inria Lille North-European associated team funding with SOFT/VUB for 2 years.

Student: Matteo Marra, collaboration with Eliza Gonzalez Boix.

Marcus Denker gave a lecture at VUB in October 2019.

9.4.3. Inria International Partners

9.4.3.1. 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

- Abdelhakim Bouremel [University of Skikda, Algeria, Oct 2019]
- Mohamad Chakroun [Mar 2019]
- Victor Martín Dias [University of Chile, Chile, until Sep 2019]
- Christopher Fuhrman [École de technologie supérieure de Montréal, Canada, until Sep 2019]
- Yann-Gaël Guéhéneuc [Concordia University, Canada, from Apr 2019 until May 2019]
- Sebastijan Kaplar [University of Novi Sad, Serbia, Aug 2019]
- Manuel Leuenberger [University of Bern, Switzerland, from Sep 2019 until Nov 2019]
- Milton Mamani Torres [Object Profile SpA, Chile, Aug 2019]
- Nina Medic [University of Novi Sad, Serbia, from Jun 2019 until Jul 2019]
- Hayatou Oumarou [University of Maroua, Cameroun, from Sep 2019 until Oct 2019]
- Giuseppe Antonio Pierro [University of Cagliari, Italy, until July 2019]
- Gordana Rakic [University of Novi Sad, Serbia, from Nov 2019]
- Moussa Saker [University Badji Mokhtar-Annaba, Algeria, from Dec 2019]

9.5.1.1. Internships

- Dayne Lorena Guerra Calle [Inria, from Feb 2019 until Aug 2019]
- Chia Yu Li [Inria, until Jul 2019]
- Iona Thomas [Centrale Lille, from Jul 2019 until Aug 2019]
- Oleksandr Zaitsev [Inria, until Feb 2019]

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 2019, the yearly Smalltalk conference that brings together research and industry http://www.esug.org/

10.1.2. Scientific Events: Selection

10.1.2.1. Chair of Conference Program Committees

- Anne Etien has been PC chair of IWST since 2015.
- Anne Etien has been PC chair of Sattose 2019.

- Santiago Bragagonolo: IWBOSE 2020 CFP Manager, Chair (workshop on blockchain software engineering).
- Stéphane Ducasse: PC-chair International Conference on Software Reuse 2020.

10.1.2.2. Member of the Conference Program Committees

- Marcus Denker has been a member of ICOOLPS 2019.
- Steven Costiou has been member of IWST 2019.
- Anne Etien has been member of BENEVOL 2019, IWor 2019, VISSOFT 2019, ICPC 2019, CIEL 2019, GDRGPL 2019, Modelwards Program Committees.
- Guillermo Polito has been PC of IWST 2019
- Guillermo Polito has been PC of Meta Workshop 2019.
- Vincent Aranega was in the PC of IWST 2019.

10.1.2.3. Reviewing Activities

- Vincent Aranega reviewed for MODELSWARD19.
- Damien Pollet reviewed for ICSOFT.

10.1.3. Journal

10.1.3.1. Member of the Editorial Boards

- Anne Etien has been a member of the editorial board for the Special issue on Dynamic Languages of *Science of Computer Programming* journal.
- Anne Etien has been editor for *CEUR* journal vol 2510.

10.1.3.2. Reviewing Activities

- Anne Etien was reviewer for *Journal of Systems and Software*.
- Steven Costiou was reviewer for *Science of Computer Programming*.
- Guillermo Polito was reviewer for *Journal IEEE Transactions on Industrial Informatics*.
- Damien Pollet was reviewer for *Science of Computer Programming*.
- Gordana Rakic was reviewer for the 3rd call for *The Programming Journal* (and Programming 2020 conference).
- Gordana Rakic was reviewer for the *Computing and Informatics* journal.

10.1.4. Invited Talks

Steven Costiou: Keynote ICCR2019 6th International Conference on Cloud and Robotics (Remote and live debugging of IOT applications: tools for researchers and developers.)

10.1.5. Scientific Expertise

- Anne Etien: expert for the French government on Research Tax Credit.
- Anne Etien: expert for the ANRT to evaluate a CIFRE file.
- Steven Costiou: program committee of the GDR-GPL 2020 challenges.

10.1.6. Research Administration

- Steven Costiou: representative of the post-doc researchers of the CRIStAL lab for the 2019 HCERES evaluation.
- Anne Etien is animator of the Software Engineering thematic Group of the CRIStAL lab since 2019.
- Anne Etien is elected at the committee of Inria Lille - Nord Europe center since 2016.

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

Master: Marcus Denker, 2 hours, Advanced Reflection. MetaLinks, VUB Brussels, Belgium.

Master: Steven Costiou, Fondamentaux du debugging, 20h, M2, Université de Lille

Master: Steven Costiou, Fondamentaux du debugging, 12h eTD, M2, Université de Bretagne Occidentale

Master: Steven Costiou, Introduction au C, 10h, M1 + M2, Polytech-Lille

Master: Steven Costiou, IOT, 13h, M2, Polytech-Lille

Master: Steven Costiou, IOT, 8h, M2, Université de Picardie Jules Verne

Licence: Steven Costiou, Systèmes d'information, 36h, L3, Université de Bretagne Occidentale

Master: Christophe Demarey, Intégration continue, 16 EdTD, M2, Université de Lille, France

Master: Bragagnolo Santiago, Robotique avec ROS, 36h TD, M2, ISEN

Master: Bragagnolo Santiago, Introduction a la blockchain, 6h TD, M2, Université de Lille

Licence: Bragagnolo Santiago, Programmation par objet, 26.3h TD, Polytech-Lille, France

Licence: Anne Etien, Bases de données, 30h, L3, Polytech-Lille, France

Licence: Anne Etien, Programmation par objet, 40h, L3, Polytech-Lille, France

Licence: Anne Etien, Bases de données, 30h, M1, Polytech-Lille, France

Master: Anne Etien, Test et Maintenance, 10h, M1, Polytech-Lille, France

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

Master: Anne Etien, Système d'information objet, 10h, M1, Polytech Lille, France

Master: Anne Etien, Bases de données Avancées, 17h, M1, Polytech-Lille, France

Licence: Vincent Aranega, Programmation Python, 42h, niveau (L1), Université de Lille, France

Master: Vincent Aranega, Génie Logiciel, 42h, niveau (M1), Université de Lille, France

Master: Vincent Aranega, Paradigme de Programmation par la Pratique, 40h, niveau (M1), Université de Lille, France

Master: Vincent Aranega, Paradigme de Programmation par la Pratique, 15h, niveau (M1), Université de Lille, France

Licence: Vincent Aranega, Programmation C, 42h, niveau (L2), Université de Lille, France

Licence: Vincent Aranega, Programmation avancée, 32h, niveau (L3), Polytech-Lille, France

Licence: Nicolas Anquetil, Principes des Système d'Exploitation, 30h, L2, IUT-A, Université de Lille, France

Licence: Nicolas Anquetil, Conception OO Avancée, 34h, L2, IUT-A, Université de Lille, France

Licence: Nicolas Anquetil, Modélisation Mathématique, 36h, L2, UT-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, Graphes & Languages, 64h, L2, IUT-A, Université de Lille, France

Licence: Nicolas Anquetil, Suivi de Stages, 15h, L2, IUT-A, Université de Lille, France

Licence: Nicolas Anquetil, Interfaces Hommes-Machines, 32h, L2, IUT-A, Université de Lille, France

Licence: Benoît Verhaeghe, Structure de données, 22 EdTD, L3, Polytech Lille, France

Licence: Benoît Verhaeghe, Système d'exploitation, 20 EdTD, L2, Université de Montpellier, France

Licence: Benoît Verhaeghe, Fondamentaux Architecture et Systèmes d'Exploitation, 15 EdTD, L3, Polytech Montpellier, France

Master: Benoît Verhaeghe, Architecture Logicielles, 25 EdTD, M1, Polytech Lille, France

Master: Benoît Verhaeghe, Évolution et restructuration, 9 EdTD, M2, Université de Montpellier, France

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

License: Damien Pollet, Programmation objet en Java, 140h, L3, IMT Lille-Douai, France

License: Damien Pollet, Systèmes numériques, 18h, L3, IMT Lille-Douai, France

Master: Damien Pollet, Technologies des systèmes d'informations, 28h, M1, IMT Lille-Douai, France

Master: Damien Pollet, Ingénierie du logiciel, 7h, M2, IMT Lille-Douai, France

Master: Damien Pollet, Algorithmes pour les réseaux, 22h, M2, IMT Lille-Douai, France

Licence: Julien Delplanque, Structure de données, 18 EdTD, L3, Polytech Lille, France

Licence: Julien Delplanque, Base de données relationnelles, 16 EdTD, L3, Polytech Lille, France

Master: Julien Delplanque, Base de données, 12 EdTD, M1, Polytech Lille, France

Master: Julien Delplanque, Informatique Industrielle 2, 12 EdTD, M1, Polytech Lille, France

Master: Stéphane Ducasse. Pharo, 20h, University of Prague, Czech Republic

Master: Stéphane Ducasse, Advanced Design, 16h, M1, Université Brest, France

Licence: Stéphane Ducasse, Interpreters, 24h, L3, Université de Lille, France

Master: Stéphane Ducasse, Pharo, 2h, M1, Polytech Lille, France

Master: Stéphane Ducasse, Advanced Design, 2h, M1, ENS Paris Sud, France

Master: Stéphane Ducasse, Advanced Design, 20h, M1, Université de Picardie, France

Master: Stéphane Ducasse, Advanced Design, 20h, M2, Tunis ENSI, Tunisia

Master: Stéphane Ducasse, Advanced Design, 9h, M1, Université de Brest, France

Master: Stéphane Ducasse, Modelisation, 18h, M1, Paris Pantheon Sorbonne, France

Licence: Carolina Hernández Phillips, Algorithmique et Programmation, 12 EdTD, L3, IMT Lille Douai, France

E-learning

Pharo Mooc, 7 weeks, Licences and Master students

Pedagogical resources: *TinyBlog: Develop your First Web App with Pharo* [16] and *Pharo with Style* [17]

10.2.2. Supervision

PhD: Jason Lecerf, *Designing Language-Agnostic Code Transformation Engines*, 26 Nov 2019, CEA, Thierry Goubier, Stéphane Ducasse

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

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

10.2.3. Juries

Dynamic program analysis for suggesting test improvements to developers, Oscar Luis Vera Perez, KTH Royal Institute of Technology, (Sweden), 17/12/2019.

Adaptation non-anticipée de comportement : application au déverminage de programmes en cours d'exécution, Steven Costiou, Université de Bretagne Occidentale, Brest (France), 26/11/2018.

Modélisation et évaluation de la sécurité des parcours d'authentification Youssou Ndaye, Université Rennes 1, France, 10/12/19

Challenges in the collaborative evolution of a proof language and its ecosystem Théo Zimmermann, Université de Paris, France, 12/12/19

ARIANE: Automated Re-documentation to Improve software Architecture uNderstanding and Evolution, Alexandre Le Borgne, IMT Mines Alès, France, 12/12/19.

Contribution à la conception d'un Système de Recommandation dédié à la réutilisation de composants logiciels, Brice Evrard Tarehy, Université de Fianarantsoa, Madagascar, Jan. 2020.

10.3. Popularization

10.3.1. Articles and contents

- Olivier Auverlot, Stéphane Ducasse, Luc Fabresse. *TinyBlog: Créer votre première application web avec Pharo* [15].
- Olivier Auverlot, Stéphane Ducasse, Luc Fabresse. *TinyBlog: Develop your First Web App with Pharo* [16].
- Stéphane Ducasse. *Pharo with Style* [17].

10.3.2. Education

- A MOOC for Pharo is online (Stéphane Ducasse).
  http://mooc.pharo.org
- The Pharo Consortium was part of GSOC (Google Summer of Code) again in 2019. Oleksandr Zaitsev was co-organizer. RMoD mentored some of the Students.

10.3.3. Interventions

- Multiple public Pharo Sprints in Lille.
  https://association.pharo.org/events
- RMOD co-organized and participated at ESUG 2019.
  http://www.esug.org/wiki/pier/Conferences/2019

10.3.4. Internal action

- 30 Minutes de Sciences, Inria Lille - Nord Europe. 25 octobre 2019: *Hard bugs and how to track them with Unanticipated Debugging*, Steven Costiou.

11. Bibliography

Publications of the year

International Conferences with Proceedings

[1] N. ANQUETIL, A. ETIEN, G. ANDREO, S. DUCASSE. *Decomposing God Classes at Siemens*, in "International Conference on Software Maintenance and Evolution (ICSME)", Cleveland, United States, October 2019, https://hal.inria.fr/hal-02395836

[2] J. DELPLANQUE, S. DUCASSE, G. POLITO, A. P. BLACK, A. ETIEN. *Rotten Green Tests*, in "ICSE 2019 - International Conference on Software Engineering", Montréal, Canada, May 2019, https://hal.inria.fr/hal-02002346

[3] J. DELPLANQUE, S. DUCASSE, O. ZAITSEV. *Magic Literals in Pharo*, in "IWST19 - International Workshop on Smalltalk Technologies Cologne", Köln, Germany, August 2019, https://hal.inria.fr/hal-02266137

[4] T. DUPRIEZ, G. POLITO, S. COSTIOU, V. ARANEGA, S. DUCASSE. *Sindarin: A Versatile Scripting API for the Pharo Debugger*, in "Proceedings of the 15th ACM SIGPLAN International Symposium on Dynamic Languages", Athens, Greece, October 2019, https://arxiv.org/abs/1909.03658 [*DOI :* 10.1145/3359619.3359745], https://hal.archives-ouvertes.fr/hal-02280915

[5] D. GUERRA CALLE, J. DELPLANQUE, S. DUCASSE. *Exposing Test Analysis Results with DrTests*, in "International Workshop on Smalltalk Technologies", Cologne, Germany, August 2019, https://hal.inria.fr/hal-02404040

[6] C. HERNÁNDEZ PHILLIPS, G. POLITO, L. FABRESSE, S. DUCASSE, N. BOURAQADI, P. TESONE. *Challenges in Debugging Bootstraps of Reflective Kernels*, in "IWST19 - International workshop on Smalltalk Technologies", Cologne, Germany, August 2019, https://hal.archives-ouvertes.fr/hal-02297710

[7] P. MISSE-CHANABIER, V. ARANEGA, G. POLITO, S. DUCASSE. *Illicium A modular transpilation toolchain from Pharo to C*, in "IWST19 — International Workshop on Smalltalk Technologies", Köln, Germany, August 2019, https://hal.archives-ouvertes.fr/hal-02297860

[8] G. A. PIERRO, H. S. C. ROCHA. *The Influence Factors on Ethereum Transaction Fees*, in "2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)", Montreal, Canada, IEEE, May 2019, pp. 24-31 [*DOI :* 10.1109/WETSEB.2019.00010], https://hal.inria.fr/hal-02403098

[9] G. POLITO, P. TESONE, E. MIRANDA, D. SIMMONS. *GildaVM: a Non-Blocking I/O Architecture for the Cog VM*, in "International Workshop on Smalltalk Technologies", Cologne, Germany, August 2019, https://hal.archives-ouvertes.fr/hal-02379275

[10] T. ROGLIANO, G. POLITO, P. TESONE. *Towards easy program migration using language virtualization*, in "IWST19 - International Workshop on Smalltalk Technologies", Cologne, Germany, August 2019, https://hal.archives-ouvertes.fr/hal-02297756

[11] B. VERHAEGHE, N. ANQUETIL, S. DUCASSE, A. SERIAI, L. DERUELLE, M. DERRAS. *Migrating GWT to Angular 6 using MDE*, in "SATToSE 2019 - 12th Seminar on Advanced Techniques & Tools for Software Evolution", Bolzano, Italy, July 2019, https://hal.inria.fr/hal-02304301

[12] B. VERHAEGHE, A. ETIEN, N. ANQUETIL, A. SERIAI, L. DERUELLE, S. DUCASSE, M. DERRAS. *GUI Migration using MDE from GWT to Angular 6: An Industrial Case*, in "SANER 2019 - 26th edition of the IEEE International Conference on Software Analysis, Evolution and Reengineering", Hangzhou, China, February 2019, https://hal.archives-ouvertes.fr/hal-02019015

[13] B. VERHAEGHE, C. FUHRMAN, L. GUERROUJ, N. ANQUETIL, S. DUCASSE. *Empirical Study of Programming to an Interface*, in "Automated Software Engineering (ASE 2019)", San Diego, United States, November 2019, https://hal.inria.fr/hal-02353681

Conferences without Proceedings

[14] B. VERHAEGHE, A. ETIEN, S. DUCASSE, A. SERIAI, L. DERUELLE, M. DERRAS. *Migration de GWT vers Angular 6 en utilisant l'IDM*, in "CIEL 2019 - 8ème Conférence en Ingénierie du Logiciel", Toulouse, France, June 2019, https://hal.inria.fr/hal-02304296

Scientific Books (or Scientific Book chapters)

[15] O. AUVERLOT, S. DUCASSE, L. FABRESSE. *TinyBlog: Créer votre Première Application Web avec Pharo*, Square Bracket Associates, 2019, forthcoming, https://hal.archives-ouvertes.fr/hal-02297691

[16] O. AUVERLOT, S. DUCASSE, L. FABRESSE. *TinyBlog: Develop your First Web App with Pharo*, Square Bracket Associates, April 2019, forthcoming, https://hal.archives-ouvertes.fr/hal-02297688

[17] S. DUCASSE. *Pharo with Style*, Square Bracket Associates, April 2019, https://hal.archives-ouvertes.fr/hal-02299550

[18] S. DUCASSE, H. S. C. ROCHA, S. BRAGAGNOLO, M. DENKER, C. FRANCOMME. *SmartAnvil: Open-Source Tool Suite for Smart Contract Analysis*, in "Blockchain and Web 3.0: Social, economic, and technological challenges", Routledge, February 2019, https://hal.inria.fr/hal-01940287

Research Reports

[19] M. DENKER, N. ANQUETIL, S. DUCASSE, A. ETIEN, D. POLLET. *Project-Team RMoD 2018 Activity Report*, Inria, February 2019, https://hal.inria.fr/hal-02006630

References in notes

[20] 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. 231–240, http://rmod.lille.inria.fr/archives/papers/Anqu00b-ICSM-GraphsConcepts.pdf

[21] 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

[22] 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

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

[24] 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

[25] 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

[26] 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

[27] 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

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

[29] 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

[30] 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

[31] 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

[32] 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

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

[34] 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

[35] K. FISHER, J. REPPY. *Statically typed traits*, University of Chicago, Department of Computer Science, December 2003, n° TR-2003-13

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

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

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

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

[40] 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

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

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

[43] 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

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

[45] 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

[46] 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

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

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

[49] 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

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

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

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

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

[54] 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

[55] 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

[56] 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

[57] 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

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

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

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

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

[62] 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