2025 Activity Report Project-Team EVREF

Marcus Denker, Steven Costiou, Stephane Ducasse, Anne Etien, Guillermo Polito, Larisa Safina, Imen Sayar

▶ **To cite this version:**

Marcus Denker, Steven Costiou, Stephane Ducasse, Anne Etien, Guillermo Polito, et al.. 2025 Activity Report Project-Team EVREF. Inria. 2026. ⟨hal-05658973⟩

HAL Id: hal-05658973

https://inria.hal.science/hal-05658973v1

Submitted on 16 Jun 2026

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

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

Copyright - All rights reserved

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Inria

2025 *Activity Report*

RESEARCH CENTRE: **Inria Centre at the University of Lille**

IN PARTNERSHIP WITH: **Université de Lille, Berger-Levrault**

Project-Team

EVREF

Reflective Evolution of Ever-running Software Systems

In collaboration with Centre de Recherche en Informatique, Signal et Automatique de Lille

Project-Team EVREF

Creation of the Project-Team: 2023 April 01

Each year, Inria research teams publish an Activity Report presenting their work and results over the reporting period. These reports follow a common structure, with some optional sections depending on the specific team. They typically begin by outlining the overall objectives and research programme, including the main research themes, goals, and methodological approaches. They also describe the application domains targeted by the team, highlighting the scientific or societal contexts in which their work is situated. The reports then present the highlights of the year, covering major scientific achievements, software developments, or teaching contributions. When relevant, they include sections on software, platforms, and open data, detailing the tools developed and how they are shared. A substantial part is dedicated to new results, where scientific contributions are described in detail, often with subsections specifying participants and associated keywords. Finally, the Activity Report addresses funding, contracts, partnerships, and collaborations at various levels, from industrial agreements to international cooperations. It also covers dissemination and teaching activities, such as participation in scientific events, outreach, and supervision. The document concludes with a presentation of scientific production, including major publications and those produced during the year.

Keywords

Computer sciences and digital sciences

A2. – Software sciences

A2.1. – Programming Languages

A2.1.3. – Object-oriented programming

A2.1.10. – Domain-specific languages

A2.1.12. – Dynamic languages

A2.2.1. – Static analysis

A2.2.3. – Memory management

A2.2.5. – Run-time systems

A2.2.7. – Adaptive compilation

A2.2.8. – Code generation

A2.5. – Software engineering

A2.5.3. – Empirical Software Engineering

A2.5.4. – Software Maintenance & Evolution

A2.5.5. – Software testing

A2.6. – Infrastructure software

A2.6.3. – Virtual machines

A2.6.4. – Ressource management

A4.4. – Security of equipment and software

Other research topics and application domains

B6. – IT and telecom

B6.1. – Software industry

B6.1.1. – Software engineering

B6.1.2. – Software evolution, maintenance

B6.5. – Information systems

Contents

Project-Team EVREF **1**

1 Team members, visitors, external collaborators **5**

2 Overall objectives **7**

3 Research program **7**
3.1 Research Axes within EVREF . . . 8
3.2 Axis 1 – Evolution of Ever-running Systems . . . 8
3.3 Axis 2 – New Generation Tools for Daily Tasks . . . 9
3.4 Axis 3 – A Generative Approach to Modular and Versatile Virtual Machines . . . 9

4 Application domains **10**
4.1 Programming Languages and Tools . . . 10
4.2 Software Reengineering . . . 10

5 Social and environmental responsibility **10**
5.1 Footprint of research activities . . . 10
5.2 Impact of research results . . . 10
5.2.1 Environment . . . 10
5.2.2 Social . . . 10

6 Highlights of the year **11**
6.1 Awards . . . 11

7 Latest software developments, platforms, open data **11**
7.1 Latest software developments . . . 11
7.1.1 Moose . . . 11
7.1.2 Pharo . . . 11
7.1.3 PharoVM . . . 12
7.1.4 Druid . . . 12
7.1.5 OpalCompiler . . . 13
7.1.6 Illimani Memory Profiler . . . 13
7.1.7 Chest . . . 13
7.1.8 Debugging Spy . . . 13
7.1.9 Phex . . . 14
7.1.10 Sindarin . . . 14
7.1.11 Scopeo . . . 14
7.1.12 Ranger . . . 15
7.1.13 Mutalk . . . 15
7.1.14 HeapFuzzer . . . 15
7.1.15 Compleshion . . . 16
7.1.16 Coypu . . . 16
7.1.17 Phausto . . . 16
7.1.18 Soil . . . 16
7.2 New platforms . . . 17
7.2.1 Pharo . . . 17
7.2.2 Moose . . . 17

8 New results **17**
8.1 Evolution of Ever-running Systems . . . 17
8.2 New Generation Tools for Daily Tasks . . . 18
8.3 A Generative Approach to Modular and Versatile Virtual Machines . . . 21
8.4 Crosscutting all Axis / Support . . . 21
8.5 Pharo for Live Coding Music . . . 22

9 Bilateral contracts and grants with industry **23**
9.1 Berger Levrault, France . . . 23
9.2 Thales DMS, Brest, France: Graphics . . . 23
9.3 Pharo Consortium . . . 23
9.4 Lifeware AG, Switzerland . . . 24
9.5 CIFRE Framatome, Courbevoie, France . . . 24
9.6 Open Source Collaboration with ApptiveGrid GmbH. Germany . . . 24

10 Partnerships and cooperations **24**
10.1 International research visitors . . . 24
10.1.1 Visits of international scientists . . . 24
10.1.2 Other european programs/initiatives . . . 25
10.2 National initiatives . . . 27
10.3 Regional initiatives . . . 29

11 Dissemination **30**
11.1 Promoting scientific activities . . . 30
11.1.1 Scientific events: organisation . . . 30
11.1.2 Scientific events: selection . . . 30
11.1.3 Journal . . . 31
11.1.4 Invited talks . . . 31
11.1.5 Leadership within the scientific community . . . 31
11.1.6 Scientific expertise . . . 31
11.1.7 Research administration . . . 31
11.2 Teaching - Supervision - Juries - Educational and pedagogical outreach . . . 32
11.2.1 Teaching . . . 32
11.2.2 Supervision . . . 33
11.2.3 Juries . . . 34
11.2.4 Educational and pedagogical outreach . . . 34
11.2.5 Specific official responsibilities in science outreach structures . . . 35
11.2.6 Participation in Live events . . . 35
11.2.7 Others science outreach relevant activities . . . 35

12 Scientific production **35**
12.1 Major publications . . . 35
12.2 Publications of the year . . . 36

1 Team members, visitors, external collaborators

Research Scientists

- Stéphane Ducasse [Team leader, INRIA, Senior Researcher, HDR]
- Nicolas Anquetil [UNIV LILLE, Associate Professor Detachement, HDR]
- Christophe Bortolaso [BERGER-LEVRAULT, Senior Researcher]
- Steven Costiou [INRIA, Researcher, HDR]
- Marcus Denker [INRIA, Researcher]
- Nicolas Hlad [BERGER-LEVRAULT, Researcher]
- Guillermo Polito [INRIA, Researcher, HDR]
- Larisa Safina [INRIA, ISFP]
- Anas Shatnawi [BERGER-LEVRAULT, Researcher]
- Benoit Verhaeghe [BERGER-LEVRAULT, Researcher]

Faculty Members

- Anne Etien [UNIV LILLE, Professor, HDR]
- Imen Sayar [UNIV LILLE, Associate Professor]

PhD Students

- Omar Abedelkader [INRIA]
- Nour Ayachi [INRIA]
- Valentin Bourcier [INRIA, until Oct 2025]
- Gabriel Darbord [INRIA, until Sep 2025]
- Romain Degrave [INRIA, from Nov 2025]
- Remi Dufloer [INRIA]
- Sebastian Jordan Montano [INRIA]
- Soufyane Labsari [INRIA]
- Federico Javier Lochbaum [INRIA]
- Marius Mignard [INRIA, from Sep 2025]
- Fouazi Rayane Mokhefi [INRIA]
- Nahuel Palumbo [INRIA, until Nov 2025]
- Younoussa Sow [FRAMATOME, CIFRE]
- Megha Sudheendran [INRIA, from May 2025]

Technical Staff

- Jules Boulet [INRIA, Engineer, from Sep 2025]
- Romain Degrave [INRIA, Engineer, from Jun 2025 until Oct 2025]
- Christophe Demarey [INRIA, Engineer, 60%]
- Cyril Ferlicot-Delbecque [INRIA, Engineer]
- Esteban Lorenzano [INRIA, Engineer, Pharo Consortium]
- Pablo Tesone [INRIA, Engineer, Pharo Consortium]
- Clotilde Toullec [INRIA, Engineer]

Interns and Apprentices

- Radoniaina Lucio Andry Razafindrainibe [INRIA, Intern, from Apr 2025 until Aug 2025]
- Rayane Bouzid [INRIA, Intern, from Apr 2025 until Jul 2025]
- Antoine Ciric [INRIA, Intern, from Apr 2025 until Jul 2025]
- Alexis Cnockaert [INRIA, Apprentice]
- Leo Defossez [INRIA, Apprentice, from Oct 2025]
- Leo Defossez [UNIV LILLE, Intern, from May 2025 until Aug 2025]
- Matias Nicolas Demare [INRIA, Intern, from Apr 2025 until Sep 2025]
- Enzo Demeulenaere [INRIA, Apprentice, until Sep 2025]
- Oceane Dubois [INRIA, Intern, from Apr 2025 until Jul 2025]
- Pol Durieux [INRIA, Apprentice]
- Renaud Fondeur [INRIA, Apprentice, until Aug 2025]
- Evan Joly [UNIV LILLE, Intern, from Apr 2025 until Jul 2025]
- Matija Kljajic [INRIA, Intern, until Apr 2025]
- Javier Agustin Larre Vargas [INRIA, Intern, from Dec 2025]
- Dylan Lemaire [UNIV LILLE, Intern, from Apr 2025 until Jul 2025]
- Yilin Liu [IMT LILLE DOUAI, Intern, until Feb 2025]
- Sebastian Lorenzano [INRIA, Intern, from Apr 2025 until Jul 2025]
- Ignacio Esteban Losiggio [INRIA, Intern, until Mar 2025]
- Marius Mignard [INRIA, Apprentice, until Aug 2025]
- Quentin Moutte [INRIA, Intern, from Jun 2025 until Sep 2025]
- Toky Ratolojanahary [INRIA, Intern, from Apr 2025 until Aug 2025]
- Santiago Viana [INRIA, Intern, until Feb 2025]

Administrative Assistant

- Aurore Dalle [INRIA]

Visiting Scientists

- Domenico Cipriani [Independent Researcher and Musician, three visits in 2025]
- Ghizlane El Boussaidi [ETS MONTREAL, from Feb 2025 until Jun 2025]
- Kaspar Osterbye [Independent Researcher, from Sep 2025 until Sep 2025]
- Balša Šarenac [UNIV NOVI SAD]

2 Overall objectives

The objectives of EVREF are to study and support the continuous evolution of large software systems in a holistic manner following three main axes: (1) analyses and approaches for migration and evolution of existing (legacy) systems, (2) new tools to support daily evolution, and (3) infrastructure to build language runtimes to support software evolution, new tools, frugal systems and security language features. In the context of the first axis, we propose a specific research agenda with Berger-Levrault R&D, an international software publisher headquartered in France.

Evolving large software systems is a challenge. Decades of academic research have *somehow* produced tools and platforms that help companies to maintain their systems. But keeping legacy systems active and relevant is still a really complex task. An aggravating challenge is that some of these systems can never stop (production lines, wafer production systems, auction managers, etc) and need to be updated while running at production sites. In addition, because the production environment is not the same as the development environment, the only way to spot and fix a bug is often by directly accessing software *in production, while running*.

Supporting the evolution of such *ever-running* systems is an important challenge for our industry because it must deal with more and more changing requirements and the need for dynamic adaptation.

To address this challenge EVREF works on:

1. *Analyses and approaches based on language-specific metamodels and their accompanying processes* such as test generation, semi-automated migration, or business rule identification;
2. *New generation debuggers, profilers and tools for reverse engineering*. We tackle new areas such as the support for non-functional requirements (robustness, memory consumption, ...);
3. *Language and runtime infrastructure to support evolution, green computing, security, and tooling* as a step towards self-evolvable runtimes. The EVREF approach is reflective in the sense that by controlling the underlying execution engine it will explore different facets of evolution and tooling.

3 Research program

EVREF is built around a holistic vision of the eternal software challenge. It acknowledges that we need to be able to work on different levels to support the evolution of software under different scenarios. Working on a full stack creates a situation where the team is in the position to think and propose solutions that would not be possible otherwise. The emphasis of reflection in the project title is that the axes can reflect and influence each other and can help each other in client/provider of problems or solutions.

The agenda defined with Berger-Levrault acts a reality ground for this research agenda. The evolution challenges faced by Berger-Levrault are still unresolved problems that any software company has to struggle with: testing to control migration, migration to new technology, business rule identification and software maps. These do not imply that the software is running and that migration should happen while the system is executing but they are typical scenarios.

3.1 Research Axes within EVREF

The research axes in EVREF are built to form an articulated whole around the challenges of evolution of constantly changing and running systems. The three axes are interconnected often in client relationships, *e.g.*, profilers require low-level information provided by virtual machines but virtual machines require advanced profilers. Controlling virtual machines opens the doors of many possibilities both at the level of tools but also language design for isolation or security.

- **Axis 1 – Evolution of Ever-running Systems.** This axis is about how to effectively evolve large and complex software. This covers a large spectrum of topics such as visualisation, metrics, analysis,... This includes for example migration from one language to another or between different library versions. This is within this axis that the team works in partnership with Berger-Levrault. The axis is built around the Moose platform 7.1.1 and its current redesign effort.
- **Axis 2 – New Generation Tools for Daily Tasks.** This axis is about how to offer advanced tools for everyday development: it focuses on debuggers, profilers and tools to reverse engineer code. It follows the work around debugging started in RMOD.
- **Axis 3 – A Generative Approach to Modular and Versatile Virtual Machines.** This axis is about how to improve the building of virtual machines to support their exploration and application to tools, security, green computing, ... This axis is also providing infrastructure for the other axes. The exploratory action is an important basis for this axis. In addition interactions with the Pharo consortium engineers and the use of the industry level Pharo virtual machine naturally take place.

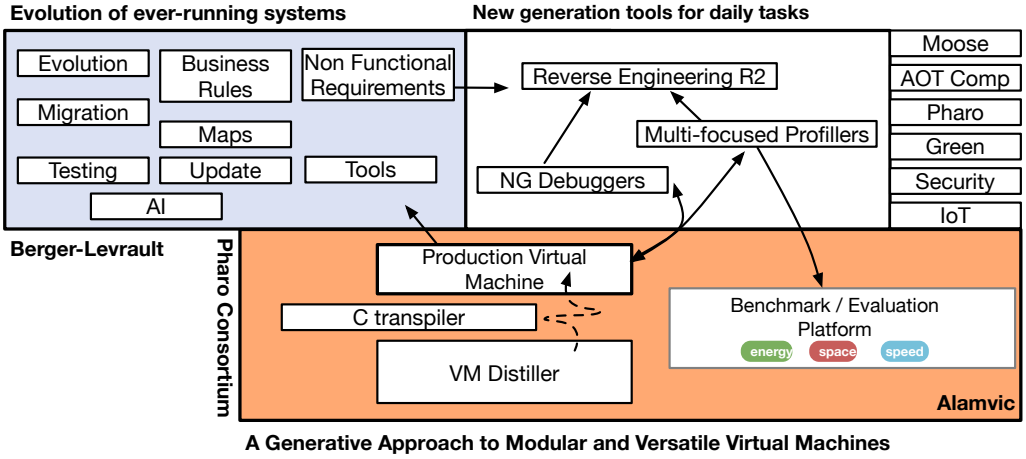

Figure 1: EVREF vision: Three interacting axes.

There are possible and welcomed overlaps between the areas covered in the interaction with Berger-Levrault: for example transpilation is the basis of the Pharo VM compilation tool chain and migration is a topic of interest for Berger-Levrault. Still we list such item in the axis because the research agenda of EVREF is larger than its interaction with Berger-Levrault. A cross-fertilisation on the same topic happen but without one taking over the others.

The third axis, *A Generative Approach to Modular and Versatile Virtual Machines*, also supports the other axes by exposing specific runtime information (such as exposing Polymorphic inline caches, possibility of instrument object creation,...) or offering the possibility to extend the virtual machines with new or modified low-level functionality. It also take into account the needs and feedback from the tool builders.

3.2 Axis 1 – Evolution of Ever-running Systems

Supporting software evolution is an important and challenging topic inherently linked to software. Indeed software models the world and the world is changing. Therefore software evolution is ineluctable. In EVREF

work on fundamental aspects of software evolution:

- **Towards automatic evolution.** We work on supporting semi-automated evolution in the case of libraries update. We extend our work and support both the library developers and their users to migrate to more recent versions by analysing past activities and learning automated rules.
- **Migration.** We enhance our metamodel-based approach of front-end migration to support interlanguage migration.
- **Non-functional requirement identification and extraction.** To help developers during their maintenance tasks we take into account non-functional requirements (NFR) and propose software maps and reverse engineering techniques to reveal such hidden software aspects.
- **NFR aware code transformations.** We extend refactorings to support domain specific and non-functional requirements.

3.3 Axis 2 – New Generation Tools for Daily Tasks

We work on tools to support developers with a focus on improving daily development tasks.

- **New Generation Debuggers** propose new debugging techniques such as object centric and back in time debuggers.
- **Multi-Layered profilers** rethink profilers and how systems are benchmarked.
- **Reverse engineering revisited** revisit reverse engineering techniques taking into account non-functional requirements such as memory consumption, security concerns and others.

3.4 Axis 3 – A Generative Approach to Modular and Versatile Virtual Machines

Virtual machines are key assets both from an engineering and research point of view. As extremely complex pieces of software (garbage collector, multi layered interpreters, speculative inliner), they raise the question of their definition, construction and validation. As a research vehicule they are keys for innovation at the level of language design, security, ever-running systems, or green computing. This axis is based on the work the team did together with the Pharo consortium and the support of the Alamvic Inria Exploratory Action led by Guillermo Polito.

Main Objective. EVREF explores how Virtual Machines are designed as a **whole**, and how they are optimised for a *large range of concerns* that include not only execution speed but also energy and space consumption, for applicability in *security, green computing, IoT and robotics*. Such research effort take place in the context of the Pharo virtual machines and its associated production chain.

- **A transpilation chain.** Based on our current architecture, we design a transpilation chain that take into account heuristics (memory, concurrency, shipset, speed).
- **Metamodeling and DSL for VM building.** VM optimisations are complex and spread over many aspects of the logic, we evaluate how such optimisations can be represented and extracted to be recomposed using a domain specific language.
- **JIT compilers and optimizers.** Building modern Just in time compilers and native dynamic optimizers is a difficult task but key to support modern language execution, we want to assess the design and architecture of alternative dynamic optimizers.
- **New Evaluation Methodologies.** A VM is a complex piece of software with adaptative behavior. We work on ways to measure performance to be able to gather actionable information.

4 Application domains

4.1 Programming Languages and Tools

Many of the results of EVREF 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 EVREF develops, is used for a very broad range of applications. The use of Pharo spans 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, Music and embedded applications. For projects done with Pharo, we refer to the Pharo Success Stories.

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 EVREF 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

5.2.1 Environment

Work on the execution environment of the Pharo programming language allows us to both improve performance and lower energy consumption. Having the expertise and responsibility over the virtual machine puts EVREF in a position where the team can easily make performance-efficiency tradeoffs where necessary. These options would not be available for EVREF if the team used a more traditional software stack.

Reengineering, in the context of software development, can be comprehended as a form of *"recycling"*. This process is akin to giving a new lease of life to existing systems. Our innovative tools enable companies to extend the lifecycle of their software systems significantly. By doing so, they can continue utilizing their current systems for a more extended period, thereby curtailing the need to invest in entirely new projects. This approach not only optimizes resource utilization but also substantially lessens the environmental footprint associated with software development. The reduced frequency of creating new software from scratch means less energy consumption and waste generation during the development process. Moreover, this practice aligns seamlessly with sustainable development goals, as it promotes efficient use of resources and minimizes the ecological impact of technological advancement. In essence, through reengineering, companies can achieve a dual objective: enhancing the longevity and functionality of their software while contributing positively to environmental conservation.

5.2.2 Social

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

- Release of Pharo 13 pharo.org
- The application for an *Inria International Chair* for Prof. James Noble was accepted. He will arrive in 2026

6.1 Awards

- Pharo : Prix science ouverte du logiciel libre de recherche, édition 2025
- IWST 2025, Gdansk, Poland: Best Paper Awards 2nd Place: *"Clean Blocks at the Opal Compiler"* [27]. Marcus Denker, Nahuel Palumbo

7 Latest software developments, platforms, open data

7.1 Latest software developments

7.1.1 Moose

Name: Moose: Software Analysis and Re-engineering Platform

Keywords: Software engineering, Meta model, Software visualisation, Parsing, Software quality, Code analysis

Scientific Description: Moose is a program manipulation platform based on a generic meta-model of programming languages.

A collection of atomic properties of programming languages (eg. an entity has a name, it can be invoked, it has a type, ...) allows to build specialized meta-models for each programming language.

The Moose analysis platform is based on these atomic properties to offer generic tools independent of the programming languages handled.

Functional Description: Moose is an open and extensible platform for software analysis and re-engineering.

It integrates language models, metrics, analysis algorithms, and visualization and navigation engines. Moose's development has been estimated at 200 man-years.

URL: `https://modularmoose.org`

Publication: hal-02972159v1

Contact: Nicolas Anquetil

Participants: Anne Etien, Nicolas Anquetil, Stephane Ducasse

Partners: Université de Berne, Sensus, Pleiad, USI, Vrije Universiteit Brussel, Berger-Levrault

7.1.2 Pharo

Name: The platform Pharo

Keywords: Live programmation objet, Reflective system, Web Application, Test, Virtual Machine Image, Object-Oriented Programming

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.

Release Contributions: Better, faster, cleaner

URL: http://www.pharo.org

Publication: hal-03687932v1

Contact: Marcus Denker

Participants: Steven Costiou, Christophe Demarey, Esteban Lorenzano, Marcus Denker, Stephane Ducasse, Guillermo Polito, Pablo Tesone, Cyril Ferlicot-Delbecque

Partners: Reveal, Inceptive, Feenk, GemTalk Systems, Greyc Université de Caen - Basse-Normandie, Yesplan, Sensus, Université de Bretagne Occidentale, École des Mines de Douai, ENSTA, Uqbar foundation Argentina, ZWEIDENKER, LifeWare, KnowRoaming, ENIT, Spesenfuchs, FINWorks, Esug, FAST, Ingenieubüro Schmidt, Projector Software, Inspired.org, High Octane, Soops, Osoco, Ta Mère SCRL, University of Yaounde 1, Software Quality Laboratory, Software Institute Università della Svizzera italiana, Universdad Nacional de Quilmes, UMMISCO IRD, Université technique de Prague

7.1.3 PharoVM

Name: Pharo Virtual Machine

Keywords: Compilation, Interpreter, Virtual Machine, Garbage Collection, Interoperability

Functional Description: The current implementation presents the following core features:

- an indirect threaded bytecode compiler using GNU extensions

- a generational scavenger garbage collector: semi-space + nursery for the young generation, a mark-compact collecting for the old generation

- a space for permanent objects that need not to be scanned by the GC

- a baseline JIT compiler that translates primitive operations using IR templates and translates bytecode methods using a simple abstract interpretation approach to reduce memory pressure (less loads/stores)

- FFI through the well-known libFFI, and support for non-blocking FFI using worker threads

URL: https://github.com/pharo-project/pharo-vm/

Contact: Guillermo Polito

7.1.4 Druid

Name: Druid Meta-Compilation Infrastructure

Keywords: Compilers, Source-to-source compiler, Optimizing compiler, Interpreter, Software engineering

Functional Description: JIT (Just-in-Time) compilers are an optimization technique often used for interpreted languages and virtual machines. They allow to spend time optimizing only frequently used code, while falling back in slower execution engines for non-frequent code. For example, the Pharo and the Java VM run on a bytecode interpreter and eventually compile machine code for methods that are frequently called.

Nowadays, the Pharo Virtual Machine is implemented in a subset of the Pharo language called Slang. The Virtual Machine developers then benefit from the high-level tools used to work with Pharo code, such as the code editors, testing frameworks and debuggers. In a later stage, the Virtual Machine code written in Slang is transpiled to C and then compiled to the target architectures.

The current Pharo JIT compiler that is part of the Virtual Machine, aka Cogit, implements an architecture based on templates of native code per bytecode. When a method is compiled, each bytecode is mapped to its corresponding template. All templates are concatenated to form a single machine code method. This architecture has as drawback that the behavior of the Pharo language is duplicated in both the bytecode interpreter and their corresponding machine code templates.

The Druid project explores the automatic generation of machine code templates from bytecode interpreters using an abstract interpreter on the existing bytecode interpreter (a meta-interpreter).

URL: https://github.com/Alamvic/druid/

Contact: Guillermo Polito

7.1.5 OpalCompiler

Keywords: Compilation, Compilers, Bytecode, Object, Object-Oriented Programming

Functional Description: Opal is the compiler that compiles source code to bytecode in Pharo.

Opal uses the Parser and AST of Pharo as input (which is used for syntax highlighting, refactoring and other tools). It does name analysis, annotating this AST before generating an Intermediate Representation (IR) with bytecode level abstractions. The IR is then used to generate bytecode and create a Pharo method.

Opal is part of Pharo and shipped with Pharo by default.

URL: http://pharo.org

Contact: Marcus Denker

7.1.6 Illimani Memory Profiler

Name: Illimani Memory Profiler

Keywords: Non volatile memory, Smalltalk, Object-Oriented Programming

Functional Description: Illimani is a library of memory profilers. It provides a memory allocation profiler and a finalization profiler. The allocation profiler gives you information about the allocation sites and where the objects where produced in your application. The finalization profiler gives you information about how much time did the objects lived, and about how many GC cycles (both scavenges and full GC) they survived.

URL: https://github.com/jordanmontt/illimani-memory-profiler

Contact: Sebastian Jordan Montano

7.1.7 Chest

Keywords: Debug, Object-Oriented Programming

Functional Description: Chest is a Pharo tool providing an API and a Graphical User Interface (GUI) to store and access objects from anywhere in the Pharo system.

URL: https://github.com/pharo-spec/Chest

Contact: Steven Costiou

Participants: Steven Costiou, Adrien Vanegue

7.1.8 Debugging Spy

Keyword: Debug

Functional Description: A tool to spy on debugging actions for research experiments conducted with Pharo.

URL: https://github.com/Pharo-XP-Tools/DebuggingSpy

Contact: Steven Costiou

Participants: Adrien Vanegue, Steven Costiou

7.1.9 Phex

Name: PHaro EXperience toolbox

Keyword: Debug

Functional Description: A tool to define, build, drive and conduct empirical experiments with Pharo.

URL: https://github.com/Pharo-XP-Tools/Phex

Contact: Steven Costiou

Participant: Steven Costiou

7.1.10 Sindarin

Keywords: Object-Oriented Programming, Software engineering, Debug

Functional Description: Sindarin is a versatile and interactive debugger scripting API for object-oriented programming languages. Sindarin is designed to help building dedicated debugging tools targeting specific problems or domains. To do this, Sindarin attaches to a running process then exposes stepping and introspection operations to control, manipulate and observe that process' execution. It simplifies the creation of personalized debugging scripts by providing an AST-based API, thus also proposing different stepping granularity over the debugging session. Once written, scripts are extensible and reusable on other scenario, and can be used to build more complex debugging tools.

URL: https://github.com/pharo-spec/ScriptableDebugger

Publications: hal-04850901, hal-02280915

Contact: Steven Costiou

Participants: Steven Costiou, Adrien Vanegue, Stephane Ducasse, Guillermo Polito

7.1.11 Scopeo

Keywords: Debug, Object-Oriented Programming

Functional Description: Scopeo is an omniscient debugger that allows developers to ask questions in the form of queries that collect objects and events related to those objects. Scopeo allows developers to save subsets of a query's results so that they can be reused as subjects for new, more refined queries. This supports the refinement of hypotheses that developers make during debugging. Thanks to the omniscient backend that Scopeo relies on (Seeker), Scopeo provides the ability to navigate back and forth in the execution and, more specifically, to jump to any point in time where a collected object-related event was triggered.

URL: https://github.com/scopeo-project/ScopeoExampleERA

Publication: hal-04627606

Contact: Valentin Bourcier

Participants: Valentin Bourcier, Steven Costiou

7.1.12 Ranger

Keywords: Symbolic testing, Software testing, Concolic Execution, Interpreter, Compilation

Functional Description: Modern language implementations using Virtual Machines feature diverse execution engines such as byte-code interpreters and machine-code dynamic translators, a.k.a. JIT compilers. Validating such engines requires not only validating each in isolation, but also that they are functionally equivalent. Tests should be duplicated for each execution engine exercising the same execution paths on each of them.

Ranger presents a novel automated testing approach for virtual machines featuring byte-code interpreters. Ranger uses concolic meta-interpretation:

it applies concolic testing to a byte-code interpreter to explore all possible execution interpreter paths and obtain a list of concrete values that explore such paths. it then uses such values to apply differential testing on the VM interpreter and JIT compiler.

This solution is based on two insights: (1) both the interpreter and compiler implement the same language semantics and (2) interpreters are simple executable specifications of those semantics and thus promising targets to (meta-) interpretation using concolic testing. We validated it on 4 different compilers of the open-source Pharo Virtual Machine and found 468 differences between them, produced by 91 different causes, organized in 6 different categories.

URL: https://github.com/Alamvic/ranger

Contact: Guillermo Polito

7.1.13 Mutalk

Keywords: Mutation analysis, Mutation testing, Software testing

Functional Description: During the 70s, mutation testing emerged as a technique to assess the fault-finding effectiveness of a test suite. It works mutating objects' behavior and looking for tests to "kill" those mutants. The surviving mutants are the starting point to writing better tests. Thus, this technique is an interesting alternative to code coverage regarding test quality.

However, so far it is a "brute force" technique that takes too long to provide useful results. This characteristic has forbidden its widespread and practical use regardless the existence of new techniques, such as schema-based mutation and selective mutation. Additionally, there are no mutation testing tools (to our knowledge) that work on meta-circular and dynamic environments, such as Smalltalk, so compile and link time are the current technique's bottleneck.

This Smalltalk-based tool was developed at the University of Buenos Aires (Argentina) in the context of the final thesis work. The tool uses Smalltalk's dynamic and meta-programming facilities to notably reduce the time to get valuable output and help to understand and implement new tests due to its integration with the rest of the environment.

URL: https://github.com/pharo-contributions/mutalk

Contact: Guillermo Polito

7.1.14 HeapFuzzer

Keywords: Fuzzing, Memory Allocation, Garbage Collection

Functional Description: Fuzzer directement un gestionnaire de mémoire nous permet de contrôler des aspects tels que l'emplacement où les objets sont alloués, et des événements de bas niveau tels que les invocations du GC et leurs paramètres. Notre solution génère de grandes séquences d'événements aléatoires sur le tas qui exercent les algorithmes de ramassage des ordures pour générer des pannes de VM et trouver des bogues. Nous combinons le fuzzing avec un algorithme de réduction des tests qui trouve le plus petit sous-ensemble d'événements reproduisant un problème.

URL: https://github.com/Alamvic/heapFuzzer

Contact: Guillermo Polito

7.1.15 Compleshion

Name: Compleshion: a code completion framework

Keywords: Pharo, Code completion

Functional Description: Compleshion is a software architecture that supports lazzily code completion. It offers the possibility to assemble different approaches to code completion in Pharo.

URL: https://www.pharo.org

Contact: Stephane Ducasse

7.1.16 Coypu

Keywords: Music, Live Programming, Object-Oriented Programming

Functional Description: Coypu is a live coding package and domain-specific language for Pharo, designed to be easy to install, simple to learn, and fun to use. It follows the principles of:

- Iconicity : code that resembles its meaning
- Economy : minimal syntax, maximal clarity
- Synonymic Equivalence : flexibility in expression

The goal: code that feels more like natural human language.

URL: https://github.com/lucretiomsp/Coypu

Publications: hal-05341056v1, hal-05306163v1

Contact: Domenico Cipriani

7.1.17 Phausto

Keywords: Music, Object-Oriented Programming, Live Programming

Functional Description: Phausto is a library and API for Pharo that enables sound generation and audio Digital Signal Processing programming in Pharo. Phausto leverages a dynamic library accessed via Foreign Function Interface (FFI) calls within Pharo. This library processes synthesizers and effects defined in Phausto with the help of an embedded FAUST compiler, which handles real-time audio computation.

Publications: hal-04813572v1, hal-04837510v1, hal-04826894v1

Contact: Domenico Cipriani

7.1.18 Soil

Name: Soil DB

Keywords: Object-Oriented Programming, Databases

Functional Description: Soil is an object oriented database in pharo. It is transaction based having ACID transactions. It has binary search capabilities with SkipList and BTree+ indexes. It aims to be a simple yet powerful database making it easy to develop with, easy to debug with, easy to inspect,...

URL: https://github.com/ApptiveGrid/Soil

Publication: hal-04726251

Contact: Marcus Denker

Participants: Marcus Denker, Norbert Hartl

Partner: ApptiveGrid GmbH

7.2 New platforms

The team produces and maintains two large platforms: Pharo and Moose.

7.2.1 Pharo

Pharo pharo.org - Pharo is a dynamic reflective language and its environment 7.1.2.

In 2025, Pharo was a recipient of the Prix science ouverte du logiciel libre de recherche.

Participants: Pablo Tesone, engineer consortium, 100% , Esteban Lorenzano, engineer consortium, 100% , Christophe Demarey, engineer Inria, 80% , Stéphane Ducasse, researcher, 25% , Guillermo Polito, researcher, 50% , Marcus Denker, researcher, 50% , Steven Costiou, researcher, 30% , Martin Dias, engineer on contract, 100%, Chile , Carolina Hernandez, engineer on contract, 12 months 50%, Chile .

7.2.2 Moose

Moose modularmoose.org - Moose is meta environment to build analyses and tools of software systems 7.1.1.

Participants: Nicolas Anquetil, researcher, 40% , Anne Etien, Professor, 40% , Imen Sayar, MCF, 40% , Laria Safina, ISFP, 40% , Clotilde Toullec, engineer, 100% , Cyril Ferlicot-Delbecque, engineer on contract, 100% .

8 New results

We present the results of the year for the three axis of EVREF.

8.1 Evolution of Ever-running Systems

Participants: Marius Mignard, Steven Costiou, Nicolas Anquetil, Anne Etien, Soufyane Labsari, Imen Sayar, Benoit Verhaeghe, Nicolas Hlad, Anas Shatnawi, Younoussa Sow, Stéphane Ducasse, Nour Ayachi.

Analysing Python Machine Learning Notebooks with Moose

Machine Learning (ML) code, particularly within notebooks, often exhibits lower quality compared to traditional software. Bad practices arise at three distinct levels: general Python coding conventions, the organizational structure of the notebook itself, and ML-specific aspects such as reproducibility and correct API usage. However, existing analysis tools typically focus on only one of these levels and struggle to capture ML-specific semantics, limiting their ability to detect issues. We present Vespucci Linter, a static analysis tool with multi-level capabilities, built on Moose and designed to address this challenge. Leveraging a metamodeling approach that unifies the notebook's structural elements with Python code entities, our linter enables a more contextualized analysis to identify issues across all three levels. We implemented 22 linting rules derived from the literature and applied our tool to a corpus of 5,000 notebooks from the Kaggle

platform. The results reveal violations at all levels, validating the relevance of our multi-level approach and demonstrating Vespucci Linter's potential to improve the quality and reliability of ML development in notebook environments. [25]

Service Extraction from Object-Oriented Monolithic Systems: Supporting Incremental Migration

Migrating large monolithic systems to service-based architecture is a complex process, mainly due to the difficulty of extracting reusable functionality from tightly coupled components. To support this, Service Identification techniques have been proposed to decompose monoliths into service candidates. Implementing these service candidates requires significant restructuring efforts. To address this complexity and build confidence in the target architecture, prior research recommends using an incremental migration approach where services are extracted one at a time. However, incremental migration has been poorly explored in the literature and lacks dedicated tool support. Thus, we explore the idea of a tool-assisted service extraction to support incremental migration, where one service is extracted at each increment. [11]

Enhancing AI-Generated Code Accuracy: Leveraging Model-Based Reverse Engineering for Prompt Context Enrichment

Large Language Models (LLMs) have shown considerable promise in automating software development tasks such as code completion, understanding, and generation. However, producing high-quality, contextually relevant code remains a challenge, particularly for complex or domain-specific applications. We present an approach to enhance LLM-based code generation by integrating model-driven reverse engineering to provide richer contextual information. Our findings indicate that incorporating unit tests and method dependencies significantly improves the accuracy and reliability of generated code in industrial projects. In contrast, simpler strategies based on method signatures perform similarly in open-source projects, suggesting that additional context is less critical in such environments. These results underscore the importance of structured input in improving LLM-generated code, particularly for industrial applications. [12]

Migrating Esope to Fortran 2008 using model transformations

Legacy programming languages such as FORTRAN 77 still play a vital role in many industrial applications. Maintaining and modernizing these languages is challenging, especially when migrating to newer standards such as Fortran 2008. This is exacerbated in the presence of legacy proprietary extensions on such legacy languages, because their semantics are often based on old context (limits of legacy language, domain logic,...). We present an approach for automatically migrating FORTRAN 77 with a proprietary extension, named Esope, to Fortran 2008. We introduce a tool that converts Esope source code to Fortran 2008. While supporting readability of the generated code, we want to maintain the level of abstraction provided by Esope. Our method uses model-driven engineering techniques, with transformations to generate a target model from which we export easy-to-read Fortran 2008 source code. We discuss the advantages, limitations, and maintainability considerations of our approach and provide insights into its scalability and adaptability to evolving requirements. [19]

From Textual Descriptions to Code: A Filtering Approach for Locating Business Rules

Business rules form the backbone of enterprise applications, capturing organizational policies, legal constraints, and the decision logic that governs business processes. In legacy systems, these rules are often hidden in poorly documented code, scattered across multiple modules, and entangled with technical details. When changes are needed, developers must locate and update the relevant code fragments, a process that is time-consuming and error-prone. We propose a fully automated approach designed to assist developers in selecting the portions of code that need to be modified following a change in a business rule. The approach reduces the search space of candidate methods likely to implement the affected rule. Starting from a textual description of the rule, our approach combines natural language processing with code analysis to filter methods and retain those relevant for implementing the necessary modifications. We evaluate our solution on a real-world codebase and demonstrate its usefulness in guiding developers during software evolution tasks. [5]

8.2 New Generation Tools for Daily Tasks

Participants: Omar Abedelkader, Stéphane Ducasse, Guillermo Polito, Gabriel Darbord, Benoit Verhaeghe, Anne Etien, Nicolas Anquetil, Balša Šarenac, Domenico Cipriani, Federico Javier Lochbaum, Remi Dufloer, Imen Sayar, Steven Costiou, Valentin Bourcier, Nicolas Hlad.

Package-Aware Approach for Repository-Level Code Completion in Pharo

Pharo offers a sophisticated completion engine based on semantic heuristics, which coordinates specific fetchers within a lazy architecture. These heuristics can be recomposed to support various activities (e.g., live programming or history usage navigation). While this system is powerful, it does not account for the repository structure when suggesting global names such as class names, class variables, or global variables. As a result, it does not prioritize classes within the same package or project, treating all global names equally. We present a new heuristic that addresses this limitation. Our approach searches variable names in a structured manner: it begins with the package of the requesting class, then expands to other packages within the same repository, and finally considers the global namespace. We describe the logic behind this heuristic and evaluate it against the default semantic heuristic and one that directly queries the global namespace. Preliminary results indicate that the Mean Reciprocal Rank (MRR) improves, confirming that package-awareness completions deliver more accurate and relevant suggestions than the previous flat global approach. [20]

A Multi-Language Tool for Generating Unit Tests from Execution Traces

Legacy software systems often lack extensive testing, but are assumed to behave correctly after years of bug fixes and stable operation. Migrating or modernizing these systems is challenging because there is little support for preventing regressions. Test carving addresses this problem by generating unit tests based on the current behavior of the system, treating it as an implicit oracle. We present Modest, a multi-language tool that generates unit tests by carving them from execution traces. Modest processes method calls, including their receivers, arguments, and results, to recreate these invocations as unit tests. Its model-based approach allows it to support multiple languages. We detail how it can be extended to handle additional languages. Modest aims to generate tests that are human-readable and maintainable over time. To achieve this, it reconstructs values as source code rather than relying on deserialization. We evaluate Modest by generating tests for both Java and Pharo applications. [8]

Building Blocks for Object-Oriented Refactoring Engines

Refactorings are behavior-preserving source code transformations that have become integral to modern Integrated Development Environments (IDEs) and code editors, significantly enhancing software development practices. Our goal is to facilitate the creation of custom refactorings and support users in developing their own. To achieve this, we identify the essential building blocks required for creating composite, modular refactorings.

We propose a set of abstractions that enable the development of a robust refactoring engine. These abstractions include an initial set of transformations, program primitives, and an API for precondition checking and static analysis necessary to create new refactorings. We demonstrate how standard refactorings can be effectively composed using these building blocks, providing concrete examples to illustrate their application. [29]

Pharo Sound Design: Adding Auditory Feedback to a Live Programming Environment

We present a prototype for integrating auditory cues into the Pharo Integrated Development Environment (IDE) to support workflow awareness and explore multimodal interaction. To achieve this, we use the synthesis capabilities of the Phausto library and API, which handles digital signal processing through an embedded Faust compiler. This allows us to attach cues to user-facing events without modifying the original source code. We describe a set of event-driven auditory cues and the rationale behind their design. We then describe the components required for instrumentation and consistent cue playback. This approach provides a practical foundation for further experimenta tion with sound in live programming environments [7]

Divergence-Driven Debugging: Understanding Behavioral Changes Between Two Program Versions

Daily software changes present an inherent risk of introducing bugs. To understand such bugs, it is crucial

to understand why specific changes introduced the bugs. This is difficult and time-consuming. We propose Divergence-Driven Debugging, an approach using a debugger for detecting execution divergences between two versions of the same program. Developers can explore both versions' execution in full and compare them based on the detected divergences. We evaluate our approach on a bug scenario and show that our debugger highlights the live state and behavior required to understand why the changes produce the bug and to identify its root cause. [9]

Towards better assessing performance effectiveness: A first analysis with mutation testing

Performance benchmarking is a crucial tool for evaluating software efficiency. Unlike behavioral tests, where Mutation testing and Test Coverage propose metrics to measure test quality, there are no methodologies for evaluating the quality of benchmarks. Coverage provides insights into execution but does not necessarily correlate with performance bugs. We propose an initial approach to use performance mutation to detect performance bugs, introducing a benchmark evaluation tool. We propose to assess the effectiveness of benchmarks by measuring their capacity to find performance issues. We explore a methodology that evaluates the quality of benchmarks based on mutation testing. We introduce artificial performance bugs into programs, and we assess the benchmark's ability to detect them. We present a first experiment using this tool, showing its preliminary results, where we measure the sensitivity of benchmarks to catch artificial performance bugs. We also provide a systematic approach to validate their effectiveness in finding performance issues. We introduce an understanding of benchmark quality and offer insights into improving benchmark measurement. [24]

Empirically Evaluating the Impact of Object-Centric Breakpoints on the Debugging of Object-Oriented Programs

To investigate the impact of object-centric breakpoints on the debugging process, we devised and conducted a controlled experiment with 81 developers who spent an average of 1 hour and 30 minutes each on the study. The experiment required participants to complete two debugging tasks using debugging tools with vs. without object-centric breakpoints. We found no significant effect from the use of object-centric breakpoints on the number of actions required to debug or the effectiveness in understanding or fixing the bug. However, for one of the two tasks, we measured a statistically significant reduction in debugging time for participants who used object-centric breakpoints, while for the other task, there was a statistically significant increase. Our analysis suggests that the impact of object-centric breakpoints varies depending on the context and the specific nature of the bug being addressed. In particular, our analysis indicates that object-centric breakpoints can speed up the process of locating the root cause of a bug when the bug can be replicated without needing to restart the program. We discuss the implications of these findings for debugging practices and future research. [6]

GitProjectHealth: an Extensible Framework for Git Social Platform Mining

Git social platforms (such as Gitlab, Github, or Bitbucket) provide insight into a team's workflow. Mining Software Repositories (MSR) provides methods and tools to extract data from these platforms. However, most tools lack connectivity and extensibility across multiple platforms. Moreover, they rarely connect to other project management platforms such as Jira. We introduce GitProjectHealth (GPH), a framework to extract data from any Git repositories and social platforms. GPH is implemented inside a model-driven engineering framework in Pharo smalltalk, facilitating its extension to other social platforms. We demonstrate GPH features over 3 open-source organizations: Eclipse, MooseTechnology and Microsoft; as well as Berger-Levrault, a closed-source company. We extracted their activity to build distributions of commits by user and to determine which types of ticket were associated with each merge request. [10]

Even Lighter Than Lightweight: Augmenting Type Inference with Primitive Heuristics

Type inference, as a technique of automatic deduction of types in programming languages, plays an important role in code correctness, maintainability, and performance optimization. In dynamically typed languages, type inference presents significant challenges due to their flexible, runtime-oriented typing mechanisms. We explore a novel set of primitive heuristics designed to augment type inference in Pharo. We demonstrate that even minimal hints, such as method naming conventions and collection patterns, can produce meaningful improvements in inferred type coverage. [28]

8.3 A Generative Approach to Modular and Versatile Virtual Machines

Participants: Guillermo Polito, Stéphane Ducasse, Pablo Tesone, Nahuel Palumbo.

Are Abstract-interpreter Baseline JITs Worth it? An Empirical Evaluation through Metacompilation

Baseline JIT compilers need to compile early and as fast as possible, while still performing optimizations. One powerful technique to write fast baseline JIT compilers is abstract interpretation. Several implementations of this technique exist in practice, implementing in a single pass optimizations such as register allocation, constant propagation, and instruction scheduling. However, although they share the same technique, all these implementations vary in the exact optimizations performed, their internal design and further implementation details (e.g., the implementation language and framework). Thus, it is challenging to understand and isolate the benefits of the technique by simply studing these existing implementations.

Understanding the real impact of compile-time abstract interpreters requires isolating performance differences and experimenting with different variations of the same implementation, which demands extensive engineering work. We propose to analyse the impact of abstract interpreters through metacompilation. We use metacompilation as a means to (a) reduce the experimentation effort and (b) to produce compiler variants that are comparable, reducing implementation noise.

We implemented our solution to generate several JIT compiler variants for the Pharo VM. We describe the adaptations required in the metacompilation framework to target both abstract interpreters and direct translators, in combination with Static Type Prediction optimizations.

Our benchmarks show that compile-time abstract interpreters, on average, reduce the emitted machine code size by 12% and increase execution speed by 10%, up to 30%, without increasing JIT compilation overhead, compared to direct translators. [13]

Meta-compilation of Baseline JIT Compilers with Druid

Virtual Machines (VMs) combine interpreters and just-in-time (JIT) compiled code to achieve good performance. However, implementing different execution engines increases the cost of developing and maintaining such solutions. JIT compilers based on meta-compilation cope with these issues by automatically generating optimizing JIT compilers. This leaves open the question of how meta-compilation applies to baseline JIT compilers, which improve warmup times by trading off optimizations.

We present Druid, an ahead-of-time automatic approach to generate baseline JIT compiler frontends from interpreters. Language developers guide meta-compilation by annotating interpreter code and using Druid's intrinsics. Druid targets the meta-compilation to an existing JIT compiler infrastructure to achieve good warm-up performance.

We applied Druid in the context of the Pharo programming language and evaluated it by comparing an autogenerated JIT compiler frontend against the one in production for more than 10 years. Our generated JIT compiler frontend is 2× faster on average than the interpreter and achieves on average 0.7× the performance of the handwritten JIT compiler. Our experiment only required changes in 60 call sites in the interpreter, showing that our solution makes language VMs easier to maintain and evolve in the long run. [4]

8.4 Crosscutting all Axis / Support

Participants: Fouazi Rayane Mokhefi, Stéphane Ducasse, Luc Fabresse, Pablo Tesone, Steven Costiou, Marcus Denker, Nahuel Palumbo, Benoit Verhaeghe.

Static Escape Analysis in Pharo: Towards Minimizing Object Allocations

Object-oriented programming idioms, such as boxing numbers or using design patterns like builders and commands, often create numerous short-lived objects. These objects put pressure on garbage collectors and are ideal candidates for static optimizations such as object inlining. Escape analysis identifies these short-lived objects, offering insights for future optimizations-both manual and automatic. Traditionally, escape analysis has been applied to statically typed languages. However, dynamically-typed languages introduce additional uncertainty, particularly due to highly polymorphic call sites.

We present escapha, the first implementation of a context-sensitive, flow-insensitive, interprocedural escape analysis for the dynamically typed language Pharo. We applied escapha to various Pharo packages, and it successfully identified instances of short-lived object creation and potential locations for optimization. Our approach shows that static analysis is able to find a small amount but relevant optimizable allocation sites. We found 280 candidates for a call graph depth of 10 on 24 000 methods. [26]

It's Alive! What a Live Object Environment Changes in Software Engineering Practice

Tools shape our mind. This is why it is important to have extensible and flexible tools that developers can adapt to their needs. In addition, coding in the abstract (thinking about what objects will look like) adds abstraction to the wrong level. In Smalltalk environments like Pharo, developers interact closely with their objects,gaining immediate feedback - not guessing how these objects will look like but just talking to them. In this article we present some powerful tools developers use in Pharo. [15]

Clean Blocks at the Opal Compiler

Higher-order languages encourage programmers to use lambda functions or closures. In Smalltalk, we see a lot of use of block closures, for example, in the collection API. Closures are expensive as they have to be created at runtime, impacting code execution efficiency. However, there are closures with code agnostic to the context where they have been defined and, thus, can be created at compile time. We present Clean Blocks, an optimization implemented in the Pharo compiler for detecting and creating closures independent of the context at compile time. We also implemented a specialization for Constant Blocks, i.e., closures that only return constant values. We evaluate the impact of this optimization, statically in a new Pharo image, and dynamically by running several benchmarks and measuring closure activations and execution time. [27]

Energy Consumption of Web Applications: Measurement Challenges in Practice

Software systems consume about 6% of global energy, and new regulations promote energy efficiency. However, developers face challenges like standardized measurement tools, integrating energy monitoring, and interpreting data meaningfully, which hinders informed decisions balancing energy efficiency and performance metrics. While most studies focus on small-scale projects, we addressed this gap with an industrial case study. We developed a methodology to assess energy impact using Dynatrace Carbon Impact and NeoLoad, enabling the evaluation of technology-driven design decisions under production-like conditions. This enabled the evaluation of technology-driven design decisions, such as frontend frameworks (Java Swing vs. Angular), backend stacks (legacy Java vs. Spring Boot), and serialization formats (JSON vs. Protobuf) under production-like conditions. Our insights highlight several design recommendations for energy-efficient software in industrial contexts. [3]

8.5 Pharo for Live Coding Music

Participants: Domenico Cipriani, Sebastian Jordan Montano, Nahuel Palumbo, Stéphane Ducasse.

Coypu: Gnawing Music On-The-Fly With Pharo

Coypu is a Pharo package for programming music on-the-fly acting as a client for an external audio generator server or for an internal DSP created with the Phausto library. Pharo is a fully open-source dynamic and reflective pure object-oriented language, based on Smalltalk-80, which includes an immersive integrated development environment. Coypu was initially developed to pair with Symbolic Sound Kyma, as a full-stack Smalltalk sound-design and live-coding environment, it was soon extended to interact with Open Sound Control (OSC) servers such as ChucK, PureData, and SuperCollider, and later expanded with MIDI capabilities. During the past year, we added an API to provide a quick connection to the SuperDirt audio engine, featuring a 'String-Oriented' syntax heavily inspired by Tidal Cycles for creating what we call Sequencers. The primary purpose of developing Coypu is to provide a pure object-oriented language designed to cultivate creative coding literacy. We believe the Smalltalk-inspired approach to constructivist learning is a gateway for newcomers, and individuals with little or no programming experience. At the same time, being both reflective and modifiable, Coypu offers unlimited possibilities to advanced users. It offers an engaging entry point not only to more concise and expressive functional and procedural programming

languages but also to the world of computational systems, algorithms, and general purpose programming. [14],[23]

Composing and Performing Electronic Music on-the-Fly with Pharo and Coypu

Coypu is a Pharo package for programming music on-the-fly acting as a client for an external audio generator server or for an internal DSP created with the Phausto library. Pharo is a fully open-source dynamic and reflective pure object-oriented language, based on Smalltalk-80, which includes an immersive integrated development environment. Coypu was initially developed to pair with Symbolic Sound Kyma, as a full-stack Smalltalk sound-design and live-coding environment, it was soon extended to interact with Open Sound Control (OSC) servers such as ChucK, PureData, and SuperCollider, and later expanded with MIDI capabilities. During the past year, we added an API to provide a quick connection to the SuperDirt audio engine, featuring a 'String-Oriented' syntax heavily inspired by Tidal Cycles for creating what we call Sequencers. The primary purpose of developing Coypu is to provide a pure object-oriented language designed to cultivate creative coding literacy. We believe the Smalltalk-inspired approach to constructivist learning is a gateway for newcomers, and individuals with little or no programming experience. At the same time, being both reflective and modifiable, Coypu offers unlimited possibilities to advanced users. It offers an engaging entry point not only to more concise and expressive functional and procedural programming languages but also to the world of computational systems, algorithms, and general purpose programming. [22]

9 Bilateral contracts and grants with industry

9.1 Berger Levrault, France

Berger-Levrault is an international software publisher headquartered in France.

EVREF is a shared team with Berger-Levrault. This includes work on software architecture, test generation, and remodularization. The collaboration started 9 years ago and resulted in two finished phd theses and three ongoing ones. We organize workshops and training sessions annually where we share work and advancements in research. Berger-Levrault is now an active contributor to the Moose software analysis platform.

Participants: Christophe Bortolaso, Nicolas Anquetil, Stéphane Ducasse, Anne Etien, Nicolas Hlad, Anas Shatnawi, Benoît Verhaeghe.

9.2 Thales DMS, Brest, France: Graphics

Thales Defence Mission Systems (Thales DMS) is the European leader and ranks third worldwide in the market for airborne and naval defence mission systems and equipment. Thales uses Pharo for prototyping and internal products. See the Support Wizard by Thales for an example.

With the Pharo Consortium, from 2023. Industrial R&D collaboration with Dr. Eric Le Pors, lead prototyping architect at Thales DMS, Brest. We work on the Pharo core graphics library.

Participants: Pablo Tesone, Stéphane Ducasse, Martin Dias.

9.3 Pharo Consortium

The Pharo Consortium was founded in 2012 and is growing constantly. (From 2012, ongoing.)

Participants: Pablo Tesone, Stéphane Ducasse, Esteban Lorenzano, Marcus Denker.

9.4 Lifeware AG, Switzerland

Lifeware is a complete, fully integrated, web based solution for the management of life insurance products. Lifeware uses Pharo for development.

In collaboration with the Pharo Consortium, we improve Pharo for the specific needs of Lifeware. For example, one goal is to be able to work with very large systems (>100K classes).

Participants: Pablo Tesone, Stéphane Ducasse, Esteban Lorenzano, Marcus Denker.

9.5 CIFRE Framatome, Courbevoie, France

Framatome is an international leader in nuclear energy.

With Framatome, we have an industrial R&D collaboration on migrating a proprietary programming language to Fortran 2003 using meta-modelisation.

Participants: Nicolas Anquetil, Stéphane Ducasse, Larisa Safina, Younoussa Sow.

9.6 Open Source Collaboration with ApptiveGrid GmbH. Germany

With ApptiveGrid we work on the shared open source project Soil 7.1.18, an object oriented database in Pharo.
In 2025, we released version 3 [31], we gave talks:

- Norbert Hartl Soil: a database for fun and profit - 26 November 2025 UK Smalltalk User Group (online) 2026-01-05
- Norbert Hartl Rhizome - Distribution in Soil ESUG 2025, Gdansk, Poland, 2025-07-02
- Marcus Denker Soil: Tutorial and Q&A" ESUG 2025, Gdansk, Poland, 2025-07-02

Participants: Norbert Hartl, Marcus Denker.

10 Partnerships and cooperations

10.1 International research visitors

10.1.1 Visits of international scientists

Other international visits to the team

Balša Šarenac

Status PhD Student

Institution of origin: University of Novi Sad

Country: Serbia

Dates: 2/06-6/06

Context of the visit: Refactoring engine

Mobility program/type of mobility: research stay

Domenico Cipriani

Status Independend Research and Musician

Institution of origin: Private

Country: Italy

Dates: 14/04-17/04, 12/09-18/09, 27/11-29/11

Context of the visit: Pharo for Music

Mobility program/type of mobility: research stay

Kaspar Osterbye

Status Researcher

Institution of origin: Private

Country: Denmark

Dates: 08/09–13/09 2025

Context of the visit: LLM

Mobility program/type of mobility: research stay

Ghizlane El Boussaidi

Status Professor

Institution of origin: École de Technologie Supérieure – Montréal

Country: Canada

Dates: 02/2025–10/2025

Context of the visit: Software Architecture Refactoring tools

Mobility program/type of mobility: Sabbatical leave

10.1.2 Other european programs/initiatives

University of Novi Sad, Serbia

Participants: Stéphane Ducasse, Balša Šarenac.

We collaborate with two groups of the University of Novi Sad lead by Gordana Rakic and Gordana Milosavljevic on improving the Refacoring features of Pharo.

Balša Šarenac visited once in 2025.

University of Zurich, Switzerland

Participants: Steven Costiou, Valentin Bourcier.

Zurich Empirical Software Engineering Team (ZEST), from 2020.
We collaborate with Alberto Bacchelli on large-scale evaluations of debugging tools. This collaboration involves 3 researchers and 2 PhD students.

Pharo for Live Coding Music

Partners: Domenico Cipriani, EVREF, MINT, GRAME, Pharo Association

Participants: Domenico Cipriani, Nahuel Palumbo, Sebastian Jordan Montaño, Stéphane Ducasse, Marcus Denker, Florent Berthaut, Enzo Demeulenaere, Oceane Dubois.

With Domenico Cipriani (Italy) we are exploring how to create a live music coding environment with Pharo. We developed better abstractions to support live coding and musical shows. The work on user interaction was done in collaboration with Florent Berthaut of the Cristal MINT Team. The Focus of 2025 was Coypu, a live coding package and domain-specific language for Pharo. Three papers have been published [23] [14] [22].
Domenico Cipriani visited the team in 2025 three times.

Events for 2025:

- February 25th: Online talk for the Audio Programmer YouTube Channel "DSP Made Accessible: Fast Plug-in Development with Phausto". Video
- April 14th: Live Sound Synthesis with Phausto at Lunprovisé - Muzzix at La Malterie, Lille (France) muzzix.info/LUN19h-Lunprovise-2345
- April 18th: Audio-Visual Live Coding Performance with TurboPhausto at Maison des Etudiants, Villneuve d'Ascq (France).
- April 17th: Workshop for Electronic Fog Collective at CSO Pedro, Padova (Italy) "Live Coding Music with Pharo and Coypu".
- May 10th+11th: Workshop for children and teenagers at Festival della Robotica, Pisa (Italy). "Introduzione al Live Coding con Pharo, Coypu e Phausto" roboticafestival.it
- May 10th: Music Performance with Pharo(Coypu) and a MIDI synthesizer. Algorave at Teatro Sant'Andrea, Pisa (Italy) . "Programmare musica al volo con Pharo". roboticafestival.it
- May 31st: Workshop at Canòdrom Barcelona (Spain), for the "International Live Coding Conference 2025""Music on-the-fly with Pharo".
- May 31st: Audiovisual Live Coding with Coypu, Phausto and Bloc at Laut Club Barcelona (Spain), for the International Live Coding Conference 2025. Galloping the Mooflod
- ESUG 2025: The Code in the Corridor - audio-visual performance with Pharo for Coypu, Phausto and Bloc - July 2nd 2025.
- ESUG 2025: (Turbo)Phausto: news from the pit lane - Talk on new music features for Pharo . July 3rd 2025.
- September 26th: Online tal for ADC-x Gather (online Audio Developer Conference) "Sound Over Boilerplate: Accessible Plug-ins Development with Phausto and Cmajor". Video

- October 10th: Workshop at ROBOT Festival, Bologna (Italy) "Creare Musica Programmando!". roboticafestival.it
- November 10th: Workshop at the Audio Developer Conference (ADC) in Bristol (UK) Programming Music and Synthesizers On-The-Fly with Pharo
- November 11th: Music performance with Pharo and ChuGL at ADC25 social event at Zero Degrees, Bristol (UK). conference.audio.dev
- November 12th: Poster session at ADC25, Bristol (UK). "Phausto: Fast and Accessible DSP Programming in Pharo".[21]

Three student projects have been supervised:

Pharo Automated Mouth a Text-To-Speech Tool for Pharo Google Summer of Code project - student: Neerja Doshi / mentors Domenico Cipriani and Nahuel Palumbo. The goal of the project is to develop a text-to-speech (TTS) tool for Pharo, leveraging Phausto's formant and subtractive synthesis capabilities as the audio backend. There is a previous work done in Squeak to take as reference. gsoc.pharo.org/pam

The PharoJamSession project conducted by Oceane Dubois (intern) mentored by Stéphane Ducasse, Domenico Cipriani and Enzo Demeulenaere A graphical environment to play with customs synthesisers and effects, modelled after Faust online playground built in Pharo with Phausto and Bloc.

Extending the Pharo Playground conducted by Antoine Ciric (intern), mentored by Marcus Denker Experiments to extend the Pharo Playground to support live music coding.

10.2 National initiatives

Project LLM4Code
Partners: Evref, R. Robbes, Labri, Diverse (From 2023).

Participants: Nicolas Anquetil , Stéphane Ducasse, Larisa Safina.

Generative AI, in particular the recent Large Language Models (LLMs), show great promise for software developments. Specialized models are now able to perform an impressive variety of programming tasks: solving programming exercises, assisting software developers, or even generating mechanized proofs. Yet, many challenges still need to be addressed to build reliable and productive LLM-based coding assistants: improving the quality of the generated code, increasing the developers' confidence in the generated code, enabling interaction with other software development tools (verification, test), and providing new capabilities (automated migration and evolution of software). The goal of this project is to leverage LLM capabilities to build code assistants that can enhance both reliability and productivity. The challenge is organized along three work packages: 1) self-improving code generation, 2) evolution of existing software, and 3) interactive tools with AI-in-the-loop.

SWHSec: Leveraging Software Heritage to Enhance Cybersecurity
Partners: EVREF, Software Heritage (From 2023 to 2025).

Participants: Cyril Ferlicot-Delbecque, Stéphane Ducasse, Imen Sayar, Anne Etien, Nicolas Anquetil.

The rise of Open Source has accelerated innovation by allowing massive reuse of a huge number of freely available software components developed by a vast community distributed around the world. This has had serious consequences on the software supply chain, with the introduction of a large number of dependencies

on components whose quality level is difficult to assess and control: they can contain vulnerabilities, and become sources of attacks on the systems that depend on them, as we saw with the Log4J incident. Recent examples of deliberately sabotaged software in response to the invasion of Ukraine have shown how the line between well-intentioned and malicious actors in the software development world is becoming increasingly blurred.

The urgency of addressing these issues is now clearly perceived, as seen for example in the May 2021 White House Executive Order, which explicitly mentions the need to "ensure and attest, to the extent possible, the integrity and provenance of open source software."

To meet this imperative, it is necessary to be able to analyze the millions of publicly available software projects, study their development history, and extract relevant information.

We are fortunate to have the Software Heritage archive, an initiative launched about 6 years ago by Inria in partnership with UNESCO, which already contains more than 12 billion unique source files from more than 180 million different origins, with all their development history.

This project brings together a group of research teams with significant expertise in software source code analysis to leverage the unprecedented resource that is the Software Heritage archive and explore the possibilities it opens up in terms of cybersecurity. New features needed to enrich the archive with security-relevant information such as component dependencies and links to known vulnerabilities will be developed, used to trace the origin and impact of vulnerabilities, and automatic detection and remediation from the patterns thus detected will be explored.
These developments will provide the basis for making Software Heritage effectively usable in industrial and cyber defense applications.

ANR JCJC Sapper
Partners: EVREF, Sigma, UQAM (Quebec) (From 2023 to 2027).

Participants: Guillermo Polito, Pablo Tesone, Jean Privat, Rémi Bardenet.

In Sapper we propose a holistic approach to reduce the cost of benchmarking. Namely, we will study how to build relevant, reproducible, and interpretable benchmark programs. We will automate the generation, selection, execution and interpretation of benchmarks by reuniting fundamental, practical, and empirical knowledge from programming language implementation, software engineering, and statistics.

École Nationale d'Ingénieurs de Tarbes

Participants: Marcus Denker.

With Cédrick Béler (ENIT/LGP/ICE) we are exploring the life-cycle (contextual time relation) of data, information, and knowledge in the context of Object-Oriented data models.

ANR JCJC OCRE
Partners: EVREF, SmArtSE (UCAQ, Quebec), UX Prototyping (Thales DMS, Brest) (From 2022 to 2024).

Participants: Steven Costiou, Valentin Bourcier, Marcus Denker.

The objectives of the OCRE project are to study the fundamental and practical limits that hinder the implementation, the evaluation, and the adoption of object-centric debugging. We propose to build the first generation of object-centric debuggers, in order to identify and evaluate its real benefits to OOP debugging.

We argue that these debuggers have the potential to drastically lower the cost (time and effort) of tracking and understanding hard bugs in OOP.

Action Exploratoire Inria: AlaMVic

Participants: Guillermo Polito, Pablo Tesone, Nahuel Palumbo.

Language Virtual Machines (VMs) are pervasive in every laptop, server, and smartphone. Industry-level VMs use highly-engineered optimization techniques, often handcrafted by experts, difficult to reproduce, replicate and change. Such optimization techniques target mostly speed improvements and are incompatible with constraints such as space and energy efficiency important in the fields of IoT or robotics. In AlaMVic1 we propose to approach VM construction using a holistic generative approach, in contrast with existing approaches that focus on speed and single VM components such as the JIT compiler. We explore how to transform handcrafted optimizations into generation heuristics, how they are applied and combined in fields such as IoT and robotics, and new methods and metrics to evaluate VMs in such fields.

ANR Profil
Partners: EVREF, Université Côte d'Azur, Centre Inria de l'Université Côte d'Azur

Participants: Anne Etien, Clotilde Toullec, Florent Jaillet, Frederic Precioso, Imen Sayar, Michel Riveill, Mireille Blay-Fornarino, Nicolas Anquetil, Philippe Collet, Stéphane Ducasse, Steven Costiou.

The machine learning community faces major challenges in the industrial construction of reusable workflows: identifying ML workflows that are embedded in code, operationalizing them, reproducing them, and ultimately enabling their reuse by third parties. Addressing these challenges is crucial to capitalizing on the considerable efforts invested across all domains that use and develop ML workflows. It must be possible to verify and validate the correctness of these mass-produced workflows. This will enable the aggregation and factorization of relevant practices, and more broadly, effective control over the maintenance of systems that integrate these models.

In this project, we adopt a software engineering approach to tackle this problem by proposing to combine model-driven engineering (in the software engineering sense), static and statistical analysis to characterize these ML workflows through models (also in the software engineering sense).

10.3 Regional initiatives

IMT Douai

Participants: Pablo Tesone, Marcus Denker, Stéphane Ducasse.

We have an ongoing close collaboration with Prof Luc Fabresse around Pharo.

In the past this included both improving the language (Pharo boot-strap) as well as applications for example in IoT (PharoIoT) and Robotics (PhaROS).

The PhDs of Pablo Tesone, Pierre Misse, Carolina Hernandez and Faouzi Rayane were joint projects with the team of IMT Douai.

We have currently one shared PhD thesis in progress: Fouazi Rayane Mokhefi , Inria / IMT started oct. 2024, Stéphane Ducasse, Luc Fabreze, Pablo Tesone.

11 Dissemination

11.1 Promoting scientific activities

11.1.1 Scientific events: organisation

- The International European Smalltalk Usergroup Conference 2025 (ESUG), Gdanzk, Poland (85 PP, 4 Days)
- International Workshop on Smalltalk Technologies (IWST) 2025 (as part of ESUG).
- GT DevX (Steven Costiou, Valentin Bourcier)

General chair, scientific chair

- Stéphane Ducasse: The International European Smalltalk Usergroup Conference 2025 (ESUG)
- Larisa Safina: 20th International Federated (Conference on Distributed Computing Techniques (DisCoTec) (Satellite Events Chair)

Member of the organizing committees

- Marcus Denker: ESUG 2025 (Member of the ESUG board)
- Stéphane Ducasse: ESUG 2025 (Member of the ESUG board)
- Guillermo Polito: International Workshop on Smalltalk Technologies (IWST) 2025
- Steven Costiou: International Workshop on Smalltalk Technologies (IWST) 2025

Member of the Conference Steering Committee

- Larisa Safina: 18th Interaction and Concurrency Experience (ICE) workshop co-located with 19th International Federated Conference on Distributed Computing Techniques (DisCoTec)

11.1.2 Scientific events: selection

Member of the conference program committees

- Marcus Denker: International Workshop on Smalltalk Technologies (IWST) 2025
- Larisa Safina: 6nd IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)
- Larisa Safina: 11th European Conference On Service-Oriented And Cloud Computing (ESOCC)
- Larisa Safina: 5th International Workshop on Agility with Microservices Programming (AMP) co-located with 19th European Conference on Software Architecture (ECSA)
- Larisa Safina: International Workshop on Smalltalk Technologies (ESUG: IWST)
- Larisa Safina: Workshop on Adaptable Cloud Architectures (WACA) co-located with 20th International Federated Conference on Distributed Computing Techniques (DisCoTec)
- Anne Etien : Belgium-Netherlands Software Evolution Workshop 2025 (BENEVOL)
- Anne Etien : International Conference on Program Comprehension (ICPC) 2026
- Steven Costiou: IEEE Working Conference on Sofware Visualization (VISSOFT) 2025.
- Steven Costiou: Workshop on Future Debugging Techniques (DEBT) 2025.

11.1.3 Journal

Member of the editorial boards

- Larisa Safina: Special Issue of the Journal of Logical and Algebraic Methods in Programming (Journal Guest Editor), ISSN 2352-2216.

Reviewer - reviewing activities

- Guillermo Polito: Reviewer for ACM TOSEM, ISSN 1049-331X.
- Steven Costiou: Reviewer for Journal of Object Technology, ISSN 1660-1769.
- Steven Costiou: Reviewer for Journal of Computer Languages, ISSN: 2665-9182.
- Anne Etien: Automated Software Engineering Journal, ISSN: 0928-8910.

11.1.4 Invited talks

Guillermo Polito: Invited Speaker International Conference FAST Smalltalks 2025

11.1.5 Leadership within the scientific community

- Stéphane Ducasse: overall Leadership of the Pharo Project.
- Larisa Safina: Ethics Committee Member of Microservices Community
- Steven Costiou and Valentin Bourcier created the DevX (gt-devx.github.io) group of the GDR Sciences du Logiciel, with Benoît Combemale from the DIVERSE team, INRIA Rennes. This working group is is dedicated to all topics related to the developer experience research area, and regroups about 30 industrial and academic members. Two national meetings were held in 2025.

11.1.6 Scientific expertise

- Anne Etien : Evaluation for the Natural Sciences and Engineering Research Council of Canada NSERC

11.1.7 Research administration

- Anne Etien : Elected as a Member of CNU (section 27).
- Anne Etien : Directrice des Études de la L3 MIAGE, Université de Lille
- Guillermo Polito: Member of CER at Inria Lille - Commission Emploi Recherche
- Steven Costiou: Staff representative at Formation Specialisée de Site (Lille)
- Steven Costiou: Member of Inria working group for researchers individual evaluation
- Marcus Denker: HAL Referent EVREF
- Marcus Denker: Corresponding author and general coordinator for the yearly Inria (RADAR) and Cristal reports of the EVREF team

11.2 Teaching - Supervision - Juries - Educational and pedagogical outreach

11.2.1 Teaching

- Master: Steven Costiou, Debugging, Université de Brest, 24hTD
- Master: Steven Costiou, Conception Objet, Polytech Lille, 22hTD
- Master: Stéphane Ducasse, Meta, Université de Lille, 12hTD
- Master: Stéphane Ducasse, Conception avancée, Université de Lille, 60hTD
- Licence: Cyril Ferlicot-Delbecque, Génie logiciel, 18hTD, L3, Université de Lille
- Licence: Anne Etien, Conception orientée objet, 18h, L3, Université de Lille
- Licence: Anne Etien, Bases de données relationnelles, 16h, L3, Université de Lille
- Licence: Anne Etien, Génie Logiciel, 27h, L3, Université de Lille
- Licence: Anne Etien, Projet, 20h, L2, Université de Lille
- Master: Anne Etien, Bases de données relationnelles, 24h, M1, Université de Lille
- Master: Anne Etien, Metamodelisation, 30h, M2, Université de Lille
- Licence: Soufyane Labsari, Génie logiciel, 18hTD, L3, Université de Lille
- Master: Marcus Denker, 2 hours, Advanced Reflection, VUB Brussels, Belgium.
- Licence: Nour Ayachi, Technologies du Web, L1, FST Université de Lille, 32h TD
- Master: Imen Sayar, Introduction à la Sécurité Informatique (ISI), Master 1, FST Université de Lille, 18h TD
- Licence: Imen Sayar, Génie logiciel (GL), L3, FST Université de Lille, 36h TD
- Licence: Imen Sayar, Programmation des systèmes (PDS), L3, FST Université de Lille, 18h TD
- Licence: Imen Sayar, Conception orientée objet (COO), L3, FST Université de Lille, 36 TD
- Licence: Imen Sayar, Projet, FST Université de Lille, L2, 48h TD
- Licence: Imen Sayar, Bases de données 1 (BDD1), L2, FST Université de Lille, 21h TD
- Licence: Imen Sayar, Bases de données 1 (BDD1), L2 info, L3 Info Math, L2 PEIP, Master 1, FST Université de Lille, 6h CM
- Licence: Imen Sayar, Bases de données 2 (BDD2), L2, FST Université de Lille, 18h TD
- Master: Guillermo Polito, Analyse et Verification de Logiciel, Université de Lille, 32h CM
- Master: Guillermo Polito, Conception et Paradigmes de Programmation par la Pratique, Université de Lille, 48hTD
- Licence: Marius Mignard, ODI, Université de Lille, 18hTD
- Licence: Larisa Safina, Métaprogrammation, construction d'interpréteurs, Université de Lille, 27h TD-TP

11.2.2 Supervision

- Defended PhD: Gabriel Darbord, Approche indépendante du langage fondée sur des métamodèles pour la génération de tests unitaires à partir de traces d'exécution, Anne Etien, and Nicolas Anquetil, December 5th 2025, [17]
- Defended PhD: Valentin Bourcier, Empirical Evaluation and Novel Design of Object-Centric Debuggers to Improve the Debugging of Object-Oriented Programs, Steven Costiou, ANR-21-CE25-0004, October 30d 2025, [16]
- Defended PhD: Nahuel Palumbo, Virtual Machine Generation Techniques, Stéphane Ducasse, Guillermo Polito. November 27th 2025, [18]
- PhD in progress: Sebastian Jordan Montaño, Memory Profiling and Instrumentation, Inria + Région HDF, started oct. 2023, Stéphane Ducasse, Guillermo Polito, Pablo Tesone.
- PhD in progress: Federico Lochbaum, Performance Test Generation, ANR JCJC, started nov. 2024, Guillermo Polito.
- PhD in Progress: Soufyane Labsari, DSL et cartes scriptables pour la cartographie de systèmes patrimoniaux, since December 2023, Inria, Anne Etien and Nicolas Anquetil
- PhD in progress: Rémi Dufloer, Echo Debugging, Inria + Région HDF, started oct. 2024, Imen Sayar, Steven Costiou, Anne Etien.
- PhD in progress: Nour Ayachi, Le traçage des règles métier dans le code source, since November 2024, Inria through the EPC with BL, Anne Etien, Nicolas Anquetil.
- PhD in progress: Fouazi Rayane Mokhefi, Inria / IMT started oct. 2024, Stéphane Ducasse. Luc Fabreze, Pablo Tesone.
- PhD in progress: Megha Sudheendran, "Experimentation with LLMs for Fortran migration", Inria, started may 2025, Nicolas Anquetil, Larisa Safina.
- PhD in progress: Omar Abedelkader Inria / Labri / Cirad started oct. 2024, Stéphane Ducasse, Romain Robbes, Oleks Zaitsev
- PhD in progress: Marius Mignard, Abstraction d'un profil de machine learning ou comment intégrer de la sémantique dans les analyses statiques, started sep. 2025, Steven Costiou , Anne Etien.
- PhD in progress: Romain Degrave, Deserialization Attacks, Inria + Région HDF, started oct. 2025, Guillermo Polito, Imen Sayar.

Apprentices

- Léo Defossez supervised by Imen Sayar - Extraction of attack call stacks in Moose
- Pol Durieux supervised by Guillermo Polito — Mutation testing
- Alexis Cnockaert supervided by Stéphane Ducasse — Refactorings

Student interns

- Quentin Moutte supervised by Stéphane Ducasse - Improving the noteTaker
- Faniry Emmanuelle RANDRIAMIHAJARIVO supervised by Stéphane Ducasse - Book testing
- Hanitriniala Nicole RAKOTOSON supervised by Stéphane Ducasse - NLP in Pharo
- Antoine Ciric supervised by Marcus Denker - Extending the Pharo Playground
- Renaud Fondeur supervised by Guillermo Polito - Improving the Slang transpiler

- Pol Durieux supervised by Guillermo Polito — Mutation testing
- Ignacio Esteban Losiggio supervised by Guillermo Polito — Table-based Polymorphic Inline Caches
- Matias Demare supervised by Guillermo Polito — Branch Optimizations
- Léo Defossez supervised by Imen Sayar - Extraction of attack call stacks in Moose
- Oceane Dubois supervised by Domenico Cipriani and Stéphane Ducasse PharoJamSession

External Students

- Pablo Herrero: *A Compression Scheme for Reflective Methods* Universidad de Buenos Aires, Master (restarted 2025). Supervised by Marcus Denker.

11.2.3 Juries

- Anne Etien: Alice Loizeau, PhD *Comprendre et concevoir avec l'erreur dans les systèmes interactifs*, 9 décembre 2025, Université de Lille, as president
- Anne Etien: Alexandre Bonvoisin, PhD *Concilier performance et utilisation efficiente des ressources matérielles: le cas des services logiciels configurables*, 5 décembre 2025, Université de Lille, as president
- Anne Etien: Hugo Monfleur, PhD *Architecture à Microservice Orientée sur les Préoccupations: Langage, Bibliothèque, Boîte à Outils et Evaluation*, 28 novembre 2025, Université de Lille, as president
- Stéphane Ducasse: Arthur Navarro,PhD *Choreography projection and extraction for the verification and the optimization of reactive distributed systems: a language-runtime continuum*, Université de Lyon, (France), 10/11/2025.
- Stéphane Ducasse: Patrick Fortier, PhD *Programming language abstractions for the Internet of Things era*, Université de Lyon, (France), 14/01/2024.
- Guillermo Polito: Adam Chadler, PhD *Collecting very large heaps with teleGC*, 11 Décembre 2025, Telecom Sud Paris, Examinateur
- Larisa Safina: Arthur Navarro, PhD: *Choreography projection and extraction for the verification and the optimization of reactive distributed systems: a language-runtime continuum*, 10/11/2025 Université de Lyon, France, as an examinator.

11.2.4 Educational and pedagogical outreach

- The MOOC *Advanced object oriented design and development with Pharo*, originally realeased in 2023, is being run by fun-mooc. 1031 registerered articipants fun-mooc.fr
- May 10th+11th: Workshop for children and teenagers at Festival della Robotica, Pisa (Italy). "Introduzione al Live Coding con Pharo, Coypu e Phausto" roboticafestival.it. Domenico Cipriani
- EVREF was involved in the Google Summer of Code (GSOC) 2025 both organizational as well as mentoring some projects summerofcode.withgoogle.com
- We organized the International School on Live Object-Oriented Programming (isLoop) 2025 Stéphane Ducasse, Marcus Denker isloop.pharo.org

11.2.5 Specific official responsibilities in science outreach structures

- Guillermo Polito is a member of the Argentinian Uqbar Foundation
- Guillermo Polito, Stéphane Ducasse and Marcus Denker are members of the Pharo Board
- Stéphane Ducasse, Marcus Denker, Pablo Tesone are members of the ESUG Board (European Smalltalk Usergroup)
- Stéphane Ducasse, Marcus Denker, Pablo Tesone are members of the Board of the Pharo User Association

11.2.6 Participation in Live events

- April 14th: Domenico Cipriani: Live Sound Synthesis with Phausto at Lunprovisé - Muzzix at La Malterie, Lille (France) muzzix.info
- April 18th: Audio-Visual Live Coding Performance with TurboPhausto at Maison des Etudiants, Villneuve d'Ascq (France). Domenico Cipriani
- May 10th: Music Performance with Pharo(Coypu) and a MIDI synthesizer. Algorave at Teatro Sant'Andrea, Pisa (Italy) . "Programmare musica al volo con Pharo". roboticafestival.it. Domenico Cipriani

11.2.7 Others science outreach relevant activities

- We are maintainig a social media presence for EVREF and Pharo. We stopped using Twitter and now are on:
  - EVREF Bluesky evref.bsky.social
  - EVREF Mastodon: @evref_inria mastodon.social
  - Pharo Bluesky: pharoproject.bsky.social
  - Pharo Mastodon: @pharoproject mastodon.social
  - Pharo LinkedIn
  - Pharo Facebook: www.facebook.com/PharoProject
- EVREF organizes Public Pharo Sprints every last Friday of the month.

12 Scientific production

12.1 Major publications

[1] V. Bourcier, P. Rani, M. I. W. Santander, A. Bacchelli and S. Costiou. 'Empirically Evaluating the Impact of Object-Centric Breakpoints on the Debugging of Object-Oriented Programs'. In: *Proc. ACM Softw. Eng.* The ACM International Conference on the Foundations of Software Engineering (FSE). Vol. 2. FSE. Trondheim (Norvège), Norway: Association for Computing Machinery, 19th June 2025, pp. 914–935. DOI: 10.1145/3715759. URL: https://hal.science/hal-04948470.

[2] G. Polito, P. Tesone, J. Privat, N. Palumbo and S. Ducasse. 'Heap Fuzzing: Automatic Garbage Collection Testing with Expert-Guided Random Events'. In: ICST 2023 - International Conference on Software Testing. Dublin, Ireland, 16th Apr. 2023. URL: https://inria.hal.science/hal-03962007.

12.2 Publications of the year

International journals

[3] L. Khrouf, A. Shatnawi, R. Rouvoy, B. Verhaeghe and B. Thiam Niang. 'Energy Consumption of Web Applications: Measurement Challenges in Practice'. In: *IEEE Software* Special Issue on Green Clean Software Sustainability (Mar. 2026). DOI: 10.1109/MS.2025.3641504. URL: https://inria.hal.science/hal-05388914. In press (cit. on p. 22).

[4] N. Palumbo, G. Polito, S. Ducasse and P. Tesone. 'Meta-compilation of Baseline JIT Compilers with Druid'. In: *The Art, Science, and Engineering of Programming* (15th Feb. 2025). URL: https://hal.science/hal-05306190 (cit. on p. 21).

International peer-reviewed conferences

[5] N. Ayachi, B. Verhaeghe, C. Fuhrman and N. Anquetil. 'From Textual Descriptions to Code: A Filtering Approach for Locating Business Rules'. In: SANER 2026 - 33rd IEEE International Conference on Software Analysis, Evolution and Reengineering. Limassol, Cyprus, 17th Mar. 2026. URL: https://hal.science/hal-05466051 (cit. on p. 18).

[6] V. Bourcier, P. Rani, M. I. W. Santander, A. Bacchelli and S. Costiou. 'Empirically Evaluating the Impact of Object-Centric Breakpoints on the Debugging of Object-Oriented Programs'. In: *Proc. ACM Softw. Eng.* FSE 2025 - The ACM International Conference on the Foundations of Software Engineering. Vol. 2. FSE. Trondheim (Norvège), Norway: Association for Computing Machinery, 19th June 2025, pp. 914–935. DOI: 10.1145/3715759. URL: https://hal.science/hal-04948470 (cit. on p. 20).

[7] D. Cipriani and G. Darbord. 'Pharo Sound Design: Adding Auditory Feedback to a Live Programming Environment'. In: *Proceedings of the 32th Journées d'Informatique Musicale*. Journées d'Informatique Musicale. Lyon, France, 23rd June 2025. URL: https://hal.science/hal-05102350 (cit. on p. 19).

[8] G. Darbord, N. Anquetil, B. Verhaeghe and A. Etien. 'A Multi-Language Tool for Generating Unit Tests from Execution Traces'. In: SANER 2025. Montréal, Canada, 4th Mar. 2025. URL: https://hal.science/hal-04841805 (cit. on p. 19).

[9] R. Dufloer, I. Sayar, A. Etien and S. Costiou. 'Divergence-Driven Debugging: Understanding Behavioral Changes Between Two Program Versions'. In: ICPC 2025 - 33rd IEEE/ACM International Conference on Program Comprehension. Ottawa, Canada, 27th Apr. 2025. URL: https://inria.hal.science/hal-05114444 (cit. on p. 20).

[10] N. Hlad, B. Verhaeghe and K. Bauvent. 'GitProjectHealth: an Extensible Framework for Git Social Platform Mining'. In: *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. 2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR). Ottawa, France: IEEE, 28th Apr. 2025, pp. 596–600. DOI: 10.1109/MSR66628.2025.00094. URL: https://hal.science/hal-05127341 (cit. on p. 20).

[11] S. Labsari, I. Sayar, N. Anquetil, B. Verhaeghe and A. Etien. 'Service Extraction from Object-Oriented Monolithic Systems: Supporting Incremental Migration'. In: 2025 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). Montréal, Canada, 4th Mar. 2025. URL: https://hal.science/hal-04951335 (cit. on p. 18).

[12] B. T. Niang, I. Alili, B. Verhaeghe, N. Hlad and A. Shatnawi. 'Enhancing AI-Generated Code Accuracy: Leveraging Model-Based Reverse Engineering for Prompt Context Enrichment'. In: *Proceedings of the 20th International Conference on Software Technologies ICSOFT*. 20th International Conference on Software Technologies. Vol. 1. 346-354. Bilbao, France: SCITEPRESS - Science and Technology Publications, 25th June 2025, pp. 346–354. DOI: 10.5220/0013570300003964. URL: https://hal.science/hal-05136135 (cit. on p. 18).

[13] N. Palumbo, G. Polito, S. Ducasse and P. Tesone. 'Are Abstract-interpreter Baseline JITs Worth it? An Empirical Evaluation through Metacompilation'. In: CGO 2026 - IEEE/ACM International Symposium on Code Generation and Optimization. Sydney, Australia, 31st Jan. 2026. URL: https://inria.hal.science/hal-05407834 (cit. on p. 21).

Conferences without proceedings

[14] D. Cipriani, S. J. Montaño and N. Palumbo. 'Coypu: Gnawing Music On-The-Fly With Pharo'. In: ICLC 2025 - International Conference on Live Coding. Barcelona, Spain, 27th Mar. 2025. URL: https://hal.science/hal-05341056 (cit. on pp. 23, 26).

[15] J. Grigera, S. Costiou, J. C. Gardey and S. Ducasse. 'It's Alive! What a Live Object Environment Changes in Software Engineering Practice'. In: IDE 2026 - 3rd International Workshop on Integrated Development Environments. Rio de Janeiro, Brazil, 14th Apr. 2026. URL: https://inria.hal.science/hal-05407907 (cit. on p. 22).

Doctoral dissertations and habilitation theses

[16] V. Bourcier. 'Empirical evaluation and novel design of object-centric debuggers to improve debugging of object-oriented programs'. Université de Lille, 30th Oct. 2025. URL: https://theses.hal.science/tel-05478011 (cit. on p. 33).

[17] G. Darbord. 'Metamodel-Based Language-Agnostic Approach to Unit Test Generation From Execution Traces'. Université de lille, 5th Dec. 2025. DOI: 10.36227/techrxiv.24449092.url:https://hal.science/hal-04344518. URL: https://hal.science/tel-05457824 (cit. on p. 33).

[18] N. Palumbo. 'DRUID - Metacompilation of Baseline JIT Compilers'. Université de Lille, 28th Nov. 2025. URL: https://hal.science/tel-05492093 (cit. on p. 33).

Reports & preprints

[19] Y. Sow, N. Anquetil, L. Brault and S. Ducasse. *Migrating Esope to Fortran 2008 using model transformations*. 20th Jan. 2026. URL: https://hal.science/hal-05468029 (cit. on p. 18).

Other scientific publications

[20] O. Abedelkader, S. Ducasse, O. Zaitsev, R. Robbes and G. Polito. *Package-Aware Approach for Repository-Level Code Completion in Pharo*. 21st Dec. 2025. URL: https://hal.science/hal-05446902 (cit. on p. 19).

[21] D. Cipriani. 'Phausto: Fast and Accessible DSP Programming in Pharo'. In: ADC 2025 - Audio Developer Conference. Bristol, UK, United Kingdom, 10th Nov. 2025. URL: https://inria.hal.science/hal-05483323 (cit. on p. 27).

[22] D. Cipriani, S. J. Montaño, N. Palumbo and S. Ducasse. *Composing and Performing Electronic Music on-the-Fly with Pharo and Coypu*. 21st Dec. 2025. URL: https://hal.science/hal-05306163 (cit. on pp. 23, 26).

[23] D. Cipriani, N. Palumbo and S. Jordan Montaño. 'Coypu: Music on-the-fly with Pharo'. In: ICLC 2025 - International Conference on Live Coding. Barcelone, Spain, 27th Mar. 2025. URL: https://hal.science/hal-05308467 (cit. on pp. 23, 26).

[24] F. Lochbaum and G. Polito. *Towards better assessing performance effectiveness: A first analysis with mutation testing*. 21st Dec. 2025. URL: https://inria.hal.science/hal-05262667 (cit. on p. 20).

[25] M. Mignard, S. Costiou, N. Anquetil and A. Etien. *Analysing Python Machine Learning Notebooks with Moose*. 21st Dec. 2025. URL: https://hal.science/hal-05281005 (cit. on p. 18).

[26] F. Mokhefi, S. Ducasse, P. Tesone and L. Fabresse. *Static Escape Analysis in Pharo: Towards Minimizing Object Allocations*. 1st Oct. 2025. URL: https://hal.science/hal-05286659 (cit. on p. 22).

[27] N. Palumbo and M. Denker. *Clean Blocks at the Opal Compiler*. 21st Dec. 2025. URL: https://hal.science/hal-05299729 (cit. on pp. 11, 22).

[28] L. Safina, J. Blizničenko and R. Pergl. *Even Lighter Than Lightweight: Augmenting Type Inference with Primitive Heuristics*. 21st Dec. 2025. URL: https://hal.science/hal-05465326 (cit. on p. 20).

[29] B. Šarenac, S. Ducasse, G. Polito and G. Rakic. *Building Blocks for Object-Oriented Refactoring Engines*. 21st Dec. 2025. URL: https://hal.science/hal-05022531 (cit. on p. 19).

Software

[30] [SW] S. Ducasse, S. Costiou, M. Denker, C. Demarey, E. Lorenzano, G. Polito, P. Tesone and C. Ferlicot-Delbecque, *Pharo* 27th Apr. 2025. LIC: MIT License. HAL: ⟨hal-05048253⟩, URL: https://hal.science/hal-05048253, VCS: https://github.com/pharo-project/pharo, SWHID: ⟨swh:1:dir:5f8c3e380b4d6cf015cc952a85fdc4f70ae9ac2d;origin=https://github.com/pharo-project/pharo;visit=swh:1:snp:ea8a748e273c242da49ba409b71055e0ec997732;anchor=swh:1:rev:f932e59c198e7de07759311579f37312f0ef60cb⟩.

[31] [SW] N. Hartl and M. Denker, *Soil* version v3, 11th July 2025. ApptiveGrid; INRIA. LIC: MIT License. HAL: ⟨hal-04726251⟩, URL: https://hal.science/hal-04726251, VCS: https://github.com/ApptiveGrid/Soil, SWHID: ⟨swh:1:dir:8254669e2559fe3e21e570568f2d91ac7016dfb5;origin=https://github.com/ApptiveGrid/Soil;visit=swh:1:snp:33b6798d7a3ff08cc7934f4904d182ff85bd5aac;anchor=swh:1:rev:047171225e67e1c202d4d2ccff0a1f8b2054251b⟩ (cit. on p. 24).