RESEARCH CENTRE

Inria Centre at the University of Lille

IN PARTNERSHIP WITH: Université de Lille, Berger-Levrault

2024 ACTIVITY REPORT

Project-Team EVREF

Reflective Evolution of Ever-running Software Systems

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

DOMAIN

Networks, Systems and Services, Distributed Computing

THEME

Distributed programming and Software engineering



Contents

Project-Team EVREF 1					
1	1 Team members, visitors, external collaborators				
2	Overall objectives 4				
3	Research program3.1Research Axes within EVREF3.2Axis 1 – Evolution of Ever-running Systems3.3Axis 2 – New Generation Tools for Daily Tasks3.4Axis 3 – A Generative Approach to Modular and Versatile Virtual Machines	4 5 6			
4	Application domains4.1Programming Languages and Tools4.2Software Reengineering	7 7 7			
5	Social and environmental responsibility 5.1 Footprint of research activities 5.2 Impact of research results 5.2.1 Environment 5.2.2 Social	7 7 7 7 7			
6	Highlights of the year 6.1 Awards	8 8			
7	7.1.7 Reflectivity 7.1.8 Chest 7.1.9 Debugging Spy 7.1.10 Phex 7.1.11 Sindarin 7.1.12 Scopeo 7.1.13 Druid 7.1.14 Ranger 7.1.15 Mutalk 7.1.16 PharoVM 7.1.17 HeapFuzzer 7.1.18 OpalCompiler 7.2.1 Pharo	8 8 9 9 9 10 10 10 11 11 11 12 13 13 14 14 14 14 14			
8	 8.1 Evolution of Ever-running Systems	14 14 16 18 19 20			

9	Bilateral contracts and grants with industry	21		
	9.1 Berger Levrault, France	21		
	9.2 Thales DMS, Brest, France: Graphics	21		
	9.3 Thales DMS, Brest, France: Debugging	21		
	9.4 Pharo Consortium	22		
	9.5 Lifeware AG, Switzerland	22		
	9.6 Dedalus	22		
	9.7 CIFRE Framatome, Courbevoie, France	22		
10 Partnerships and cooperations 22				
	10.1 International initiatives	22		
	10.1.1 Inria associate team not involved in an IIL or an international program			
	10.2 International research visitors			
	10.2.1 Visits of international scientists			
	10.2.2 Visits to international teams			
	10.3 European initiatives	24		
	10.3.1 H2020 projects	24		
	10.3.2 Other european programs/initiatives			
	10.4 National initiatives	26		
	10.5 Regional initiatives	28		
11	1 Dissemination	28		
	11.1 Promoting scientific activities	28		
	11.1.1 Scientific events: organisation			
	11.1.2 Scientific events: selection			
	11.1.3 Journal	29		
	11.1.4 Invited talks			
	11.1.5 Leadership within the scientific community			
	11.1.6 Scientific expertise			
	11.1.7 Research administration	30		
	11.2 Teaching - Supervision - Juries			
	11.2.1 Supervision	31		
	11.2.2 Juries	32		
	11.3 Popularization	33		
	11.3.1 Specific official responsibilities in science outreach structures	33		
	11.3.2 Productions (articles, videos, podcasts, serious games,)	33		
	11.3.3 Participation in Live events			
	11.3.4 Others science outreach relevant activities			
12	2 Scientific production	33		
	12.1 Major publications	33		
	12.2 Publications of the year			

Project-Team EVREF

Creation of the Project-Team: 2023 April 01

Keywords

Computer sciences and digital sciences

- A2. Software
- 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

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]
- Marcus Denker [INRIA, Researcher]
- Nicolas Hlad [BERGER-LEVRAULT, Researcher]
- Guillermo Polito [INRIA, Researcher]
- 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, from Oct 2024]
- Nour Agouf [AROLLA, until Jan 2024]
- Nour Ayachi [INRIA, from Nov 2024]
- Valentin Bourcier [INRIA]
- Gabriel Darbord [INRIA]
- Remi Dufloer [INRIA, from Dec 2024]
- Aless Hosry [INRIA]
- Sebastian Jordan Montaño [INRIA]
- Soufyane Labsari [INRIA]
- Federico Javier Lochbaum [INRIA, from Nov 2024]
- Fouazi Rayane Mokhefi [INRIA, from Oct 2024]
- Nahuel Palumbo [INRIA]
- Younoussa Sow [INRIA FRAMATOME, CIFRE]
- Iona Thomas [INRIA, until Oct 2024]

Technical Staff

- Christophe Demarey [INRIA, Engineer, 60%]
- Cyril Ferlicot-Delbecque [INRIA, Engineer]
- Esteban Lorenzano [INRIA, Engineer, Pharo Consortium]
- Hernan Federico Morales [INRIA, Engineer, until Nov 2024]
- Pablo Tesone [INRIA, Engineer, Pharo Consortium]
- Clotilde Toullec [INRIA, Engineer]

Interns and Apprentices

- Ikimath Adouke Adeoye [UNIV LILLE, Intern, from May 2024 until Jul 2024]
- Juan Ignacio Bianchi [INRIA, Intern, from Mar 2024 until Jul 2024]
- Mateo Boury [INRIA, Intern, from Apr 2024 until Aug 2024]
- Alexis Cnockaert [INRIA, Apprentice, from Nov 2024]
- Alexis Cnockaert [UNIV LILLE, from Apr 2024 until Aug 2024]
- Romain Degrave [Université de Lille, from Mar 2024 until Aug 2024]
- Enzo Demeulenaere [INRIA, Apprentice]
- Remi Dufloer [INRIA, Intern, from Mar 2024 until Aug 2024]
- Pol Durieux [INRIA, Apprentice, from Sep 2024]
- Pol Durieux [INRIA, Intern, until May 2024]
- Redwane Engels [UNIV LILLE, Intern, from Apr 2024 until Aug 2024]
- Renaud Fondeur [INRIA, Apprentice, from Nov 2024]
- Renaud Fondeur [INRIA, Intern, from May 2024 until Aug 2024]
- Angel Hely [UNIV LILLE, Intern, from Apr 2024 until Aug 2024]
- Julien Lamhene [INRIA, Intern, from May 2024 until Aug 2024]
- Yilin Liu [IMT LILLE DOUAI, Intern, until Mar 2024]
- Ignacio Esteban Losiggio [INRIA, Intern, from Oct 2024]
- Yasser Melki [INRIA, Intern, from Apr 2024 until Sep 2024]
- Marius Mignard [INRIA, Apprentice, from Dec 2024]
- Quentin Moutte [UNIV LILLE, Intern, from May 2024 until Jul 2024]
- Mohamed Oukas [UNIV LILLE, Intern, from Apr 2024 until Aug 2024]
- Sarah Tathy [INRIA, Intern, from May 2024 until Aug 2024]
- Fabio Vandewaeter [INRIA, Intern, from Apr 2024 until Jul 2024]
- Adrien Vanegue [INRIA, Apprentice, until Aug 2024]

Administrative Assistant

Aurore Dalle [INRIA]

Visiting Scientists

- Dominico Cipriani [Independent Researcher and Musician, three visits]
- Ghizlane El Boussaidi [ETS MONTREAL, from Sep 2024 until Sep 2024]
- Balša Šarenac [University of Novi Sad, from Feb 2024, four visits]

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.

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. We work on a full stack (still making progress in each area) creates a situation where the team will be 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.

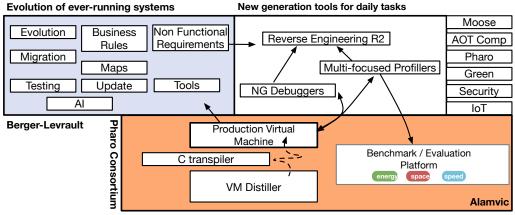
The agenda defined with Berger-Levrault acts a reality ground for this research agenda. The evolution challenges faced by Berger-Levrault are still unresolved challenges that any software company has to struggle with: testing to control migration, migration to new technology, business rule identification and software maps are key challenges. They 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 requires low-level information provided by virtual machines but virtual machines requires 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 will work 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 will naturally take place.



A Generative Approach to Modular and Versatile Virtual Machines

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 will happen but without one taking over the others.

The third axis, *A Generative Approach to Modular and Versatile Virtual Machines*, will also support 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 will 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 we will work on fundamental aspects of software evolution:

- **Towards automatic evolution.** We will work on supporting semi-automated evolution in the case of libraries update. We will 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 will 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 will 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 will extend refactorings to support domain specific and non-functional requirements.

3.3 Axis 2 – New Generation Tools for Daily Tasks

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

- New Generation Debuggers will propose new debugging techniques such as object centric and back in time debuggers.
- Multi-Layered profilers will rethink profilers and how systems are benchmarked.
- **Reverse engineering revisited** will 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 a 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 G. Polito.

Main Objective. EVREF will explore 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 will take place in the context of the Pharo virtual machines and its associated production chain.

- A transpilation chain. Based on our current architecture, we will design a transpilation chain that will 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 will 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 will 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

- EVREF organized the 2024 International European Smalltalk Usergroup Conference (ESUG) in Lille with 124 participants (8th to 11th of July 2024)
- Release of Pharo 12 (pharo.org)

6.1 Awards

• Prix 2024 de la science ouverte pour la thèse de Benoit Verhaeghe

7 New software, platforms, open data

7.1 New software

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: Stephane Ducasse

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

7.1.3 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

Participant: Marcus Denker

Partner: ApptiveGrid GmbH

7.1.4 Microdown

Name: Microdown: a minimal but extensible markdown

Keyword: Document

Functional Description: Microdown is the definition and implementation of a minimal markdown that is extensible. It integrates into the Pillar document compilation chain.

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

Contact: Stephane Ducasse

7.1.5 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

7.1.6 Pillar

Keywords: HTML, LaTeX, HTML5

Functional Description: Pillar is a markup syntax and associated tools to write and generate documentation and books.

Pillar is currently used to write several books and other documentation.

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

Contact: Stephane Ducasse

7.1.7 Reflectivity

Keywords: Object-Oriented Programming, Reflective system, Metaprogramming

Functional Description: Reflectivity is a framework for behavioral reflection.

Using Reflectivity, developers annotate Abstract Syntax Tree (AST) nodes with meta-behavior which is taken into account by the compiler to produce behavioral variations. Reflectivity brings a practical way of working at the AST level, which is a high-level representation of the source code manipulated by software developers. It enables a powerful way of dynamically add and modify behavior. Reflectivity is also a flexible mean to bridge the gap between the expression of the meta-behavior and the source code. This ability to apply unanticipated adaptation and to provide behavioral reflection led to many experiments and projects during this last decade by external users. Existing work use Reflectivity to implement reflective libraries or languages extensions, featherweight code instrumentation, dynamic software update, debugging tools and visualization and software analysis tools.Grounding. Reflectivity is actively used in research projects.

Reflectivity is part of Pharo and shipped with the default Pharo download.

URL: http://pharo.org

Publication: hal-02480136

Contact: Marcus Denker

Participants: Marcus Denker, Steven Costiou

7.1.8 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

7.1.9 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

7.1.10 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

7.1.11 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

7.1.12 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

7.1.13 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

7.1.14 Ranger

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

Functional Description: Les implémentations de langages modernes utilisant des machines virtuelles comportent divers moteurs d'exécution tels que des interprètes de code-octet et des traducteurs dynamiques de code-machine, c'est-à-dire des compilateurs JIT. Pour valider ces moteurs, il faut non seulement valider chacun d'entre eux de manière isolée, mais aussi s'assurer qu'ils sont fonctionnellement équivalents. Les tests doivent être dupliqués pour chaque moteur d'exécution en exerçant les mêmes chemins d'exécution sur chacun d'entre eux.

Ranger présente une nouvelle approche de test automatisé pour les machines virtuelles dotées d'interpréteurs de code à octets. Ranger utilise la méta-interprétation concolique :

il applique des tests concoliques à un interpréteur de code à octets afin d'explorer tous les chemins d'exécution possibles de l'interpréteur et d'obtenir une liste de valeurs concrètes qui explorent ces chemins. Il utilise ensuite ces valeurs pour appliquer des tests différentiels à l'interpréteur VM et au compilateur JIT.

Cette solution est basée sur deux idées : (1) l'interprète et le compilateur implémentent la même sémantique du langage et (2) les interprètes sont de simples spécifications exécutables de cette sémantique et donc des cibles prometteuses pour une (méta-) interprétation à l'aide de tests concoliques. Nous l'avons validé sur 4 compilateurs différents de la machine virtuelle open-source Pharo et avons trouvé 468 différences entre eux, produites par 91 causes différentes, organisées en 6 catégories différentes.

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

Contact: Guillermo Polito

7.1.15 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

7.1.16 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 - 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

7.1.17 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

7.1.18 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

7.2 New platforms

The team produces and maintains two large platforms: Pharo and Moose. See Section on New Software for a description.

7.2.1 Pharo

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

Participants:P. Tesone, engineer consortium, 100%, E. Lorenzano, engineer consortium, 100%, C. Demarey, engineer Inria, 80%, S. Ducasse, researcher, 25%, G. Polito, researcher, 50%, M. Denker, researcher, 50%, S. Costiou, researcher, 30%, H. Morales, engineer on contract, 80%, M. Dias, engineer on contract, 100%, Chile, C. 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: N. Anquetil, researcher, 40%, A. Etien, Professor, 40%, I. Sayar, MCF, 40%, L. Safina, ISFP, 40%, C. Toullec, engineer, 100%, C. 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: Imen Sayar, Anne Etien, Nicolas Anquetil, Gabriel Darbord, Benoit Verhaeghe, Iona Thomas, Stéphane Ducasse, Pablo Tesone, Guillermo Polito.

Identifying and fixing ambiguities in, and semantically accurate formalisation of, behavioural requirement

To correctly formalise requirements expressed in natural language, ambiguities must first be identified and then fixed. We focus on behavioural requirements (i.e. requirements related to dynamic aspects and phenomena). Its first objective is to show, based on a practical, public case study, that the disambiguation process cannot be fully automated: even though natural language processing (NLP) tools and machine learning might help in the identification of ambiguities, fixing them often requires a deep, applicationspecific understanding of the reasons of being of the system of interest, of the characteristics of its environment, of which trade-offs between conflicting objectives are acceptable, and of what is achievable and what is not; it may also require arduous negotiations between stakeholders. Such an understanding and consensus-making ability is not in the reach of current tools and technologies, and will likely remain so for a long while. Beyond ambiguity, requirements are often marred by various other types of defects that could lead to wholly unacceptable consequences. In particular, operational experience shows that requirements inadequacy (whereby, in some of the situations the system could face, what is required is woefully inappropriate or what is necessary is left unspecified) is a significant cause for systems failing to meet expectations. The second objective is to propose a semantically accurate behavioural requirements formalisation format enabling tool-supported requirements verification, notably with simulation. Such support is necessary for the engineering of large and complex cyber-physical and socio-technical systems to ensure, first, that the specified requirements indeed reflect the true intentions of their authors and second, that they are adequate for all the situations the system could face. To that end, we present an overview of the BASAALT (Behaviour Analysis and Simulation All Along systems Life Time) systems engineering method, and of FORM-L (FOrmal Requirements Modelling Language), its supporting language, which aims at representing as accurately and completely as possible the semantics expressed in the original, natural language behavioural requirements, and is markedly different from languages intended for software code generation. We show that generally, semantically accurate formalisation is not a simple paraphrasing of the original natural language requirements: additional elements are often needed to fully and explicitly reflect all that is implied in natural language. To provide such complements for the case study presented, we had to follow different formalisation patterns, i.e. sequences of formalisation steps. To avoid being skewed by what a particular automatic tool can and cannot do, BASAALT and FORM-L were applied manually. Still, the lessons learned could be used to specify and develop NLP tools that could assist the disambiguation and formalisation processes. However, more studies are needed to determine whether an exhaustive set of formalisation patterns can be identified to fully automate the formalisation process. [17]

Automatic Recommendations for Evolving Relational Databases Schema

Relational databases play a central role in many information systems. Their schema contains structural (e.g. tables and columns) and behavioral (e.g. stored procedures or views) entity descriptions. Then, just like for "normal" software, changes in legislation, offered functionalities, or functional contexts, impose to evolve databases and their schemas. But in some scenarios, it is not so easy to deconstruct a wished evolution of the schema into a precise sequence of operations. Changing a database schema may impose manually dropping and recreating dependent entities, or manually searching for dependencies in stored procedures. This is important because getting even the order of application of the operators can be difficult and have profound consequences. This meta-model allows us to compute the impact of planned changes and recommend additional changes that will ensure that the RDBMS constraints are always verified. The recommendations can then be compiled into a valid SQL patch actually updating the database schema in an orderly way. We replicated a past evolution showing that, without detailed knowledge of the database, we could perform the same change in 75% less time than the expert database architect. We also exemplify the use of our approach on other planned changes. [28]

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. [20]

Pharo: A Reflective Language - Analyzing the Reflective API and Its Internal Dependencies

Reflective operations are powerful APIs (Application Programming Interface) that let developers build advanced tools and architectures. Reflective operations are used for implementing tools and development environments (e.g., compilers, debuggers, inspectors) or language features (e.g., distributed systems, exceptions, proxies, aspect-oriented programming). In addition, languages are evolving, introducing better concepts, and revising practices and APIs. Since 2008 Pharo has evolved from Squeak and its reflective API has evolved accordingly, diverging consequently from the original Smalltalk reflective API. With more than 500 reflective methods identified, Pharo has one of the largest reflective feature sets ranging from structural reflection to on-demand stack reification. Those operations are often built on top of the other, creating different layers of reflective operations, from low-level to high-level ones. There is a need to understand the current reflective APIs to understand their underlying use, potential dependencies, and whether some reflective features can be scoped and optional. Such an analysis is challenged by new metaobjects organically introduced in the system, such as first-class instance variables, and their mixture with the base-level API of objects and classes. In this article, we analyze the reflective operations used in Pharo 12 and their interdependencies. We propose a classification based on their semantics and we identify a set of issues of the current implementation. Such an analysis of reflective operations in Pharo is important to support the revision of the reflective layer and its potential redesign. [8]

8.2 New Generation Tools for Daily Tasks

Participants:Iona Thomas, Stéphane Ducasse, Pablo Tesone, Steven Costiou,
Maximilian Ignacio Willembrinck Santander, Adrien Vanègue,
Valentin Bourcier, Anne Etien, Maximilian Willembrinck,
Stéphane Ducasse, Balša Šarenac, Guillermo Polito, Nicolas An-
quetil, Gabriel Darbord, Fabio Vandewaeter, Benoit Verhaeghe.

Assessing Reflection Usage with Mutation Testing Augmented Analysis

Reflection is a powerful tool that allows a program to manipulate itself during its execution. However, developers may use it to circumvent data encapsulation and method visibility modifiers. Thus, it is important to assess how much an application relies on reflection. Nonetheless, reflection is mostly incompatible with static analysis as it relies on runtime information (e.g., to determine the attribute to be accessed or the method to evaluate). These problems worsen with dynamically-typed languages, where reflective operations are polymorphic with non-reflective operations, e.g., in Pharo, array access is polymorphic with context variable modifications. We present RAPIM, an approach to study the uses of reflective APIs: it uses mutation analysis with a new mutation operator for dealing with core reflective methods. We analyze a serialization library from a developer perspective, showing the information it reveals. We evaluate our approach on a selection of five projects by comparing its performance against static analysis. When the code coverage is good, the percentage of disambiguation is three times higher. Finally, we question the relevance of polymorphism between non-reflective and reflective APIs. Out of five projects, only one uses it, for only 1.4% of potentially reflective call-sites. We argue that reflective APIs could be renamed to avoid ambiguities. [19]

Time-Traveling Object-Centric Breakpoints

Object-centric breakpoints aim to facilitate the debugging of object-oriented programs by focusing on specific objects. However, their practical application faces limitations. They often produce false positives and require developers to identify objects to debug in a running program, which is sometimes not possible

due to non-determinism. Additionally, object-centric breakpoints are difficult to build because, to the best of our knowledge, their implementations have never been abstracted from low-level concerns. The literature describes complex reflective architectures necessary for implementing these breakpoints, and their rare available implementations are language-specific. We introduce Time-Traveling Object-Centric Breakpoints (TTOCBs), a new definition and implementation of object-centric breakpoints based on Time-Traveling Queries (TTQs). TTQs are an extensible time-traveling debugging system that allows developers to explore their program executions back and forth by executing debugging queries. We argue that our query-based implementation helps to overcome the limitations of traditional object-centric breakpoints. We describe how TTOCBs assist developers in searching for objects to debug within their program executions, even in the presence of non-determinism. We illustrate how existing object-centric breakpoints from the literature can be implemented and how new ones can be created in a few steps using the TTQ abstractions and scripting API. To build breakpoints, developers need to familiarize themselves with a short API instead of learning language reflection techniques and libraries. This makes our TTOCBs independent of the underlying TTQs and debugger implementations. To evaluate our solution, we conducted an initial anecdotal user study on four example scenarios, providing evidence that debugging with TTOCBs requires fewer actions than with traditional object-centric breakpoints. We then discuss the comparison between object-centric breakpoints and TTOCBs in terms of applicability and performance. [2]

Debugging Activity Blueprint

Empirically analyzing debugging activity is notoriously difficult. In particular, aggregating data (obtained from either observation or event logging) to verify hypotheses on developers' behavior is known to be challenging. Overall, the difficulty of studying debugging activities contributes to the need for more empirical evidence on how practitioners use debuggers. We propose debugging activity blueprint as a visual tool to analyze and navigate through all the events recorded in a programming environment. Our blueprint is a polymetric view representing the interaction between debuggers and other programming tools. Our blueprint highlights the flow of a debugging activity across the tools an IDE offers. An exploratory use case over three participants and two debugging tasks indicates that our blueprint supports a fine-grained analysis of complex debugging scenarios. [9]

Scopeo: An Object-Centric Debugging Approach for Exploring Object-Oriented Programs

Debugging object-oriented programs requires understanding how objects interact with each other and how their state evolves during execution. It is an arduous activity, as there are no methods to support finding objects to debug and explore their interactions. We present Scopeo, a novel object-centric approach to explore objects and their interactions in object-oriented programs. We evaluate the performance of Scopeo and conclude its applicability to real debugging scenarios. We then open a discussion on the interest of this approach for future research into debugging and program comprehension. [10]

Time-Traveling Queries: Extensible Tools for Faster Program Comprehension

Debugging programs require program comprehension. To acquire this comprehension, developers explore the program execution, a task often performed using interactive debuggers. However, exploring a program execution through standard interactive debuggers is tedious and costly. In addition, standard debuggers are generic tools that are inflexible and difficult to use in domain-specific contexts. We propose Time-Traveling Queries (TTQs) to ease and customize program exploration. TTQs is an extensible mechanism that automatically explores program executions to collect execution data. This data is used to time-travel through execution states, facilitating the exploration of program executions. Queries can be created or extended for problem-specific or domain-specific debugging scenarios. TTQ have been successfully used on a real-world example of a bug, which shows that in practice the TTQ system is usable. To evaluate more in depth the impact of TTQs on program comprehension activities, we conducted a user study with 34 participants on program comprehension tasks. Results show that compared to traditional debugging tools, TTQs improve developers' precision (39% more correct answers) while reducing required time (27% faster to finish tasks) and effort (45% less debugging actions) when performing program comprehension tasks. [3]

Modular and Extensible Extract Method

Extract method refactoring is one of the most important refactorings in any refactoring engine because it supports developers to create new methods out of existing ones. Its importance comes with the cost of complexity since it needs to take care of many issues to produce code that is syntactically and semantically correct. Finally, their complexity often leads existing extract method refactoring to be defined in a monolithic way. Such an implementation hampers any reuse of analyses and forbids simple variations in the case of domain-specific refactorings based on extract method general idea. In this article, after describing the challenges of the analysis of Extract Method refactoring in the context of Pharo, we describe a new modular implementation. This implementation is based on the composition of elementary transformations. We validate this approach showing how it supports the natural definition of two domain-specific refactorings: Extract SetUp refactoring (for SUnit) and Extract with Pragma refactoring (for the Slang framework). [18]

A New Architecture Reconciling Refactorings and Transformations

Refactorings are behavior-preserving code transformations. They are a recommended software development practice and are now a standard feature in modern IDEs. There are however many situations where developers need to perform mere transformations (non-behavior-preserving) or to mix refactorings and transformations. Little work exists on the analysis of transformations implementation, how refactorings could be composed of smaller, reusable, parts (simple transformations or other refactorings), and, conversely, how transformations could be reused in isolation or to compose new refactorings. In a previous article, we started to analyze the seminal implementation of refactorings as proposed in the PhD of D. Roberts, and whose evolution is available in the Pharo IDE. We identified a dichotomy between the class hierarchy of refactorings (56 classes) and that of transformations (70 classes). We also noted that there are different kinds of preconditions for different purposes (applicability preconditions or behavior-preserving preconditions). In this article, we go further by proposing a new architecture that: (i) supports two important scenarios (interactive use or scripting, i.e., batch use); (ii) defines a clear API unifying refactorings and transformations; (iii) expresses refactorings as decorators over transformations, and; (iv) formalizes the uses of the different kinds of preconditions, thus supporting better user feedback. We are in the process of migrating the existing Pharo refactorings to this new architecture. Current results show that elementary transformations such as the Add Method transformation is reused in 24 refactorings and 11 other transformations; and the Remove Method transformation is reused in 11 refactorings and 7 other transformations. [7]

Modest-Pharo: Unit Test Generation for Pharo Based on Traces and Metamodels

Unit testing is essential in software development to ensure code functionality and prevent the introduction of bugs. However, challenges such as time constraints and insufficient resource allocation often impede comprehensive testing efforts, leaving software systems vulnerable to regression. To address this issue, we introduce Modest, a language-agnostic approach to unit test generation that uses metamodels and execution traces. This method ensures non-regression by replaying scenarios captured from realworld executions. We demonstrate the application of Modest to Pharo codebases by generating unit tests for two projects. [13]

8.3 A Generative Approach to Modular and Versatile Virtual Machines

Participants: Sebastian Jordan Montaño, Guillermo Polito, Stéphane Ducasse, Pablo Tesone.

Evaluating Finalization-Based Object Lifetime Profiling

Using object lifetime information enables performance improvement through memory optimizations such as pretenuring and tuning garbage collector parameters. However, profiling object lifetimes is non-trivial and often requires a specialized virtual machine to instrument object allocations and dereferences. Alternative lifetime profiling could be done with less implementation effort using available finalization mechanisms such as weak references. We study the impact of finalization on object lifetime profiling.

We built an actionable lifetime profiler using the ephemeron finalization mechanism named FiLiP. FiLiP instruments object allocations to exactly record an object's allocation time and it attaches an ephemeron to each allocated object to capture its finalization time. We show that FiLiP can be used in practice and achieves a significant overhead reduction by pretenuring the ephemeron objects. We further experiment with the impact of sampling allocations, showing that sampling reduces profiling overhead while maintaining actionable lifetime measurements. [15]

8.4 Crosscutting all Axis / Support

Participants:Sebastian Jordan Montaño, Guillermo Polito, Stéphane Ducasse,
Pablo Tesone, Aless Hosry, Nicolas Anquetil, Steven Costiou.

MethodProxies: A Safe and Fast Message-Passing Control Library

The injection of monitoring code allows for real-time observation of the program, which has proven instrumental in developing tools that assist developers with various programming tasks. In dynamic languages such as Pharo, renowned for their rich meta-programming capabilities and dynamic method dispatch, such monitoring capabilities are particularly valuable. Message-passing control techniques are commonly used to monitor program execution at the method level, involving the execution of specific code before and after each method invocation. Implementing message-passing control techniques, however, poses many challenges, notably in terms of instrumentation overhead. Additionally, it is crucial for the message-passing mechanism to be safe: i.e., to accommodate recursive and reflective scenarios to ensure that it does not alter the execution of the monitored program, which could potentially lead to infinite loops or other unintended consequences.

Over the years, numerous techniques have been proposed to optimize message-passing control. We introduce MethodProxies, a message-passing instrumentation library that offers minimal overhead and is safe. We conduct a comparison between MethodProxies and two commonly used techniques implemented in the Pharo programming language: method substitution using the run:with:in:hook and source code modification. Our results demonstrate that MethodProxies offers significantly lower overhead compared to the other two techniques and is safe against infinite recursion. [16]

MoTion: A New Declarative Object Matching Approach in Pharo

Pattern matching is an expressive way of matching data and extracting pieces of information from it. The recent inclusion of pattern matching in the Java and Python languages highlights that such a facility is more and more adopted by developers for everyday development. Other main stream programming languages also offer pattern matching capabilities as part of the language (Rust, Scala, Haskell, and OCaml), with different degrees of expressivity in what can be matched. In the meantime, in graphs, pattern matching takes a slightly different turn; it enhances the expressivity of the patterns that can be defined. Smalltalk currently offers little pattern matching capability to find specific objects inside a large graph of objects using a declarative pattern. In Pharo, the closest library to classical pattern matching that exists is the RBParseTreeSearcher, which allows to express specialized patterns over a Pharo Abstract Syntax Tree to find some inner node. The question arises of what features a flexible pattern matching languages, both in General Purpose Languages (like Java) and in declarative graph pattern matching languages. We then describe MoTion, a new pattern matching engine for Pharo smalltalk, combining all these features. We discuss some aspects of MoTion's implementation and illustrate its use with real case examples. [4]

Live Application Programming in the Defense Industry with the Molecule Component Framework

At Thales Defense Mission Systems (DMS), software products first go through an industrial prototyping phase. Prototypes are serious applications that we evaluate with our end-users during demonstrations. End-users have a central role in the design process of our products. They often ask for software modifications during demonstrations to experiment new ideas or to focus the existing design on their needs. We present how we combined Smalltalk's live-programming capabilities with software component models to obtain flexible and modular software designs in our context of live prototyping. We present Molecule, an open-source implementation of the Lightweight CORBA Component Model in Pharo. We use Molecule to build HMI systems prototypes, and we benefit from the dynamic run-time modification capabilities of Pharo during demonstrations with our end-users where we explore software designs in a lively way. Molecule is an industrial contribution to Smalltalk, as it capitalizes 20 years of usage and maturation in our prototyping activity. The Molecule framework and tools are now mature, and we started building end-user software used in production at Thales DMS. We present two such end-user software and analyze their component architecture, that are representative of how we (learnt to) build HMI prototypes. Finally, we analyze our technological decisions with regards to the benefits we sought for our industrial activity. [5]

On the Use of Statistical Machine Translation for Suggesting Variable Names for Decompiled Code: The Pharo Case

Adequately selecting variable names is a difficult activity for practitioners. In 2018, Jaffe et al. proposed the use of statistical machine translation (SMT) to suggest descriptive variable names for decompiled code. A large corpus of decompiled C code was used to train the SMT model. We present the results of a partial replication of Jaffe's experiment. We apply the same technique and methodology to a dataset made of code written in the Pharo programming language. We selected Pharo since its syntax is simple - it fits on half of a postcard - and because the optimizations performed by the compiler are limited to method scope. Our results indicate that SMT may recover between 8.9% and 69.88% of the variable names depending on the training set. Our replication concludes that: (i) the accuracy depends on the code similarity between the training and testing sets; (ii) the simplicity of the Pharo syntax and the satisfactory decompiled code alignment have a positive impact on predicting variable names; and (iii) a relatively small code corpus is sufficient to train the SMT model, which shows the applicability of the approach to less popular programming languages. Additionally, to assess SMT's potential in improving original variable names, ten Pharo developers reviewed 400 SMT name suggestions, with four reviews per variable. Only 15 suggestions (3.75%) were unanimously viewed as improvements, while 45 (11.25%) were perceived as improvements by at least two reviewers, highlighting SMT's limitations in providing suitable alternatives. [6]

Ergonomic Evaluations of Human-Machine Interfaces in the Defense Business: An Example of a Collaborative Maritime Surveillance System

Chez Thales Defense Mission System, nous impliquons les utilisateurs finaux dans la conception des systèmes logiciels, en particulier lors d'évaluations de prototypes d'Interfaces Humain-Machine. Dans les métiers de la défense, les utilisateurs finaux disposent de peu de disponibilités pour participer à ces évaluations. Leurs visites sont concentrées sur quelques jours et espacées de plusieurs mois ou années. Pour maximiser les retours, nous organisons des sessions d'évaluations ergonomiques sur des scénarios longs et complexes pour confronter nos IHM à un panorama exhaustif des situations que les utilisateurs peuvent rencontrer lors de leurs missions. Ces évaluations se déroulant sur des prototypes, il arrive que des évènements imprévus perturbent le déroulement des scénarios exécutés, par exemple des erreurs logicielles. Dans cet article, nous décrivons notre approche pour faire évoluer dynamiquement nos prototypes en cours d'évaluation et bénéficier au maximum de la présence de nos utilisateurs finaux sans interrompre les scénarios joués. [14]

8.5 Pharo for Live Coding Music

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

PHAUSTO: Embedding the Faust Compiler in the Pharo World

Phausto is a lightweight, open-source library for live-coding music, enabling sound generation and Digital Signal Processing (DSP) programming. Developed in the Pharo programming language, it incorporates the Faust compiler for robust audio capabilities, using Foreign Function Interface (FFI) calls for seamless integration. Phausto connects with platform-specific audio layers through PortAudio, offering a consistent API across operating systems. Designed for educational settings, it targets users interested in DSP, musicians, and sound artists with limited technical skills. Phausto addresses two main challenges: generating audio in Pharo applications and providing an accessible environment for programming digital musical instruments. It is easy to install and supports the latest Pharo versions, with instructions available on its GitHub repository. [11]

Phausto: Fast and Accessible DSP Programming for Sound and Music Creation in Pharo

We introduce Phausto, a library that generates sounds in Pharo programming language using Faust (Functional Audio Streams), a programming language designed to develop real-time digital signal processors (DSP).

In Phausto, DSP programs are created by the composition of Unit Generators written in a MUSIC-N style, like the ChucK programming language, or from a string containing a valid Faust program.

We present Phausto's API, implementation details and an overview of its syntax, and of Unit Generators and ToolKit elements. We also analyze the motivations behind the project and identify its target audiences. Finally, we present the conclusions drawn after one year of development and use, and outline the agenda for future work. [12]

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 8 years ago and resulted in two finished phd theses and three ongoing ones. We organize 3 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 (Tales 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.

9.3 Thales DMS, Brest, France: Debugging

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

Participants: Steven Costiou.

9.4 Pharo Consortium

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

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

9.5 Lifeware AG, Switzerland

Lifeware is a complete, fully integrated, web based solution for the management of life insurance products. In collaboration with the Pharo Consortium, we improve Pharo. The goal is to be able to work with very large systems (>100K classes).

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

9.6 Dedalus

Dedalus is a global leader in Health Software. One product is a a patient dossier for oncologists.

A collaboration started in 2021. It included a 6 month engineer position. The goals are (1) The development of a software prototype for the identification of unused functionalities within an application. (2) Analysis of the source code of the software using the open-source software platform Moose. (3) Identification of a CIFRE thesis subject on software maintenance and development.

Romain Degrave did his second internship (M2) in 2024. He migrated their home made web-service infrastructure to GWT. 600 services where automatically migrated using Moose and our software analysis tools.

Participants: Nicolas Anquetil, Stéphane Ducasse, Soufyane Labsari, Anne Etien, Romain Degrave.

9.7 CIFRE Framatome, Courbevoie, France

Framatome is an international leader in nuclear energy.

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.

10 Partnerships and cooperations

10.1 International initiatives

10.1.1 Inria associate team not involved in an IIL or an international program

SADPC

Title: Systems Analyses and Debugging for Program Comprehension

Duration: 2020 -> 2024

Coordinator: Yann-Gaël Guéhéneuc (yann-gael.gueheneuc@concordia.ca)

Partners:

· Concordia University Montréal (Canada)

Inria contact: Stéphane Ducasse

Summary: Systemic changes in the past decades have pushed software systems into all aspects of our lives, from our homes to our cars to our factories. These systems, both legacy (e.g., handling contracts for the Dept of Defense of the USA since 1958) and very recent (e.g., running the latest smart factory in France in 2019), are difficult to understand by software engineers because of their intrinsic complexity. These engineers need help understanding the systems they must adapt to the new requirements of our time. The proposed associate team considers three research directions to support the software engineers maintaining and evolving large software systems: (a) system analyses and (b) debugging for (c) program comprehension. (a) Complex algorithms often act or are perceived by software engineers as black boxes because of their intrinsic and accidental complexity, both in architecture, design, and implementation. We will develop new software analyses to support algorithm understanding. (b) Previous debugging techniques assume a unique software engineer performing a solitary debugging session. We will work on a language allowing software engineers to build their own debuggers to fit their collaborative debugging strategies. (c) Previous work on program comprehension proposed views to address one single problem at a given moment of the comprehension process. They only provide a subset of the information required by software engineers. We want to propose an approach to adapt and combine views using meta-data.

10.2 International research visitors

10.2.1 Visits of international scientists

Other international visits to the team

Ghizlane El Boussaidi

Status: Researcher

Institution of origin: ETS MONTREAL

Country: Canada

Dates: September 2024 to December 2024

Context of the visit: Associated team SADP

Mobility program/type of mobility: sabbatical

Balša Šarenac

Status: PhD Student

Institution of origin: University of Novi Sad

Country: Serbia

Dates: 5/2-16/02, 8/04-13/04, 02/07-12/07, 01/10-16/10

Context of the visit: Refactoring engine

Mobility program/type of mobility: research stay

Domenico Cipriani

Status: Independent Researcher and Musician

Institution of origin: Private

Country: Italy

Dates: 10/04 to 13/04, 14/10 to 17/10 and ESUG 8/07 to 13/07

Context of the visit: Pharo for Music

Mobility program/type of mobility: research stay

10.2.2 Visits to international teams

Research stays abroad

Guillermo Polito

Visited institution: Luxembourg Institute of Science and Technology (LIST)

Country: Luxembourg

Dates: 12-13 June 2024

Context of the visit: Visit to the software engineering team of LIST, discussions on automated testing, fuzzing and mutation

Mobility program/type of mobility: Exchange and discussion

Visited institution: Vrije Universiteit Brussel

Country: Belgium

Dates: 14 June 2024

Context of the visit: Visit to the Programming language team of LIST, discussions on debugging and instrumentation

Mobility program/type of mobility: Exchange and discussion

Valentin Bourcier

Visited institution: University of Zurich

Country: Switzerland

Dates: Aug 2024, two weeks

Context of the visit: empirical experiment and writing, FSE research paper submission

Mobility program/type of mobility: Exchange and discussion

10.3 European initiatives

10.3.1 H2020 projects

EVREF was part of the COST project CERCIRAS: Connecting Education and Research Communities for an Innovative Resource Aware Society. From 29/09/2020 to the end of the project 28/09/2024 (see www.cost.eu/actions/CA19135).

10.3.2 Other european programs/initiatives

University of Novi Sad, Serbia

Participants: Anne Etien, Nicolas Anquetil, Balša Šarenac, Stéphane Ducasse.

We collaborate with two groups of the University of Novi Sad (G. Rakic and G. Milosavljevic) on improving the Refacoring features of Pharo.

In 2024, Balša Šarenac visited four times. We published two papers: [7], [18].

University of Zurich, Switzerland

Participants: Steven Costiou, Stéphane Ducasse, Valentin Bourcier.

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

In 2024, Valentin Bourcier visited ZEST to prepare a submission for The ACM International Conference on the Foundations of Software Engineering (FSE).

Pharo for Live Coding Music

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

Participants:Dominico Cipriani, Nahuel Palumbo, Sebastian Jordan Montaño,
Stéphane Ducasse, Marcus Denker, Stéphane Sletz, Florent Berthaut,
Enzo Demeulenaere, Redwane Engels, Alexis Cnockaert.

With Dominico Cipriani (Italy) we are exploring how to create a live music coding envrionment with Pharo.

We embedded the Faust Language in with Pharo (with Stéphane Sletz of GRAME Lyon). 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.

Dominico Cipriani visited the team in 2024 three times. Notable results:

- Workshop Music with Domenico Cipriano Lille 15 November 2024
- · Concert the 17th October at the Maison des Etudiants using Pharo
- Dominico Cipriani joined an improvisation session of Muzzix 14/10 using Pharo as his instrument
- We published two papers: [12], [11]
- Experimental Audio-Visual Live Coding Performance at ESUG 2024: Riding the MOOFLOD (see video)
- Dominico Cipriani performed together with The Analogue Cops at the first Algorave at Ohm Berlin, using Pharo to compose live music on MIDI synthesiser
- Dominico Cipriani performed at System Shangai for the International Live Coding Conference 2024, using Pharo to compose music with Kyma and control visuals made with Processing

10.4 National initiatives

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.

ARCAD, Lab-STICC, Brest, France

Participants: Guillermo Polito, Pablo Tesone, Stéphane Ducasse.

We collaborate since the beginning of 2021 with the ARCAD team of the Lab-STICC in Bretagne (Prof L. Lagadec). We started at the beginning of the year with a common workshop between the two teams

looking for collaboration points. G. Polito and P. Tesone are now collaborating with the PhD of Q. Ducasse on Just-In-Time compiler technology for extensible ISA processors such as RISC-V.

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. Industrylevel 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.

La communauté du ML fait face à des défis majeurs dans la construction industrialisée de workflows réutilisables : la caractérisation de workflows de ML noyés dans les codes, leur opérationnalisation, leur reproduction, et enfin leur réutilisation par des tiers. Répondre à ces défis revêt une importance cruciale pour capitaliser sur les efforts considérables dans tous les domaines qui exploitent et développent des workflows de ML. Il doit être possible de vérifier et valider la justesse de ces workflows produits en masse. Ceci permettra l'agrégation et la factorisation des pratiques pertinentes, et plus largement la maîtrise de la maintenance des systèmes intégrant ces modèles. Dans ce projet, nous adoptons une approche de génie logiciel (GL) pour aborder cette problématique en proposant de lier ingénierie des modèles (IDM) (au sens GL), analyse statique et statistique pour caractériser ces workflows de ML par des modèles (aussi au sens GL).

10.5 Regional initiatives

IMT Douai

Participants: Pablo Tesone, Marcus Denker, Stéphane Ducasse.

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

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

The PhDs of P. Tesone, P. Misse, C. Hernandez and F. Rayane were joint projects with the team of IMT Douai.

11 Dissemination

11.1 Promoting scientific activities

11.1.1 Scientific events: organisation

- The International European Smalltalk Usergroup Conference 2024 (ESUG), Lille, France (124 PP, 4 Days)
- International Workshop on Smalltalk Technologies (IWST) 2024 (as part of ESUG).
- co-organized the nation GT DevX launch day at Rennes, December 4th (Steven Costiou, Valentin Bourcier).

General chair, scientific chair

• Stéphane Ducasse: ESUG 2024 Lille

Conference Steering Committee Member

• Larisa Safina: 17th Interaction and Concurrency Experience (ICE) workshop co-located with 18th International Federated Conference on Distributed Computing Techniques (DisCoTec)

Member of the organizing committees

- Guillermo Polito: International Workshop on Smalltalk Technologies (IWST) 2024
- Steven Costiou: International Workshop on Smalltalk Technologies (IWST) 2024
- Steven Costiou : Organisateur et Session Chair des sessions du GT Debugging, Journées Nationales du GDR-GPL 2023
- Steven Costiou : ESUG 2024 Lille: was part of organization and was responsible for student volunteers
- Marcus Denker: ESUG 2024 (Local Organizer and member of the ESUG board)
- Stéphane Ducasse: ESUG 2024 (Local Organizer and member of the ESUG board)
- Pablo Tesone: ESUG 2024 (Local Organizer and member of the ESUG board)

11.1.2 Scientific events: selection

Member of the conference program committees

- Anne Etien: IEEE Working Conference on Software Visualization (VISSOFT) 2024
- Anne Etien: 23rd Belgium-Netherlands Software Evolution Workshop 2024 (BENEVOL)
- Anne Etien: IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) ERA track, 2025
- Anne Etien: International Conference on Program Comprehension (ICPC) 2025
- Steven Costiou: Second Workshop on Future Debugging Techniques (DEBT) 2024
- Steven Costiou IEEE Working Conference on Software Visualization (VISSOFT) 2024
- Marcus Denker: RAW 2024 3rd workshop on Resource Awareness of Systems and Society
- Larisa Safina: 44th International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE)
- Larisa Safina: 5nd IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)
- Larisa Safina: 1st International Workshop on Choreographic Programming (CP 2024) co-located with 45rd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)
- Larisa Safina: 22nd International Conference on Service-Oriented Computing (ICSOC)
- Larisa Safina: International Workshop on Smalltalk Technologies (IWST) 2024
- Larisa Safina: Extended Review Committee and External Reviewers European Conference on Object-Oriented Programming (ECOOP)
- Larisa Safina: Artifact Evaluation Committee Member European Conference on Object-Oriented Programming (ECOOP)

Reviewer

- Steven Costiou: IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) 2024
- Marcus Denker: International Conference on Software Maintenance and Evolution (ICSME) 2024
- Stéphane Ducasse: International Conference on Software Reuse ICSR

11.1.3 Journal

Member of the editorial boards

• Larisa Safina: Journal Guest Editor, Special Issue of the *Journal of Logical and Algebraic Methods in Programming*

Reviewer - reviewing activities

- Marcus Denker: Elsevier Journal of Computer Languages COLA
- Marcus Denker: Journal Computer Science and Information Systems COMSIS
- Larisa Safina: Springer Nature Computational Science Nat. Comput. Sci.
- Guillermo Polito: Journal Computer Science and Information Systems COMSIS
- Guillermo Polito: ACM Transactions on Software Engineering and Methodology TOSEM

11.1.4 Invited talks

HCERES evaluation of CRIStAL:

- Sebastian Jordan Montaño: Memory Profiling in Pharo
- Gabriel Darbord: Automatic Test Generation
- Guillermo Polito: Interpreter-guided JIT Compiler Unit Testing

11.1.5 Leadership within the scientific community

- Stéphane Ducasse: overall Leadership of the Pharo Project.
- Steven Costiou is leader of the GT Debugging working group of the CNRS GDR GPL. This working group aims to gather any researcher, engineer or GDR team interested in software debugging problems. (from 2020) GDR GPL
- Steven Costiou and Valentin Bourcier are working with Benoît Combemale from the DIVERSE team, INRIA Rennes, to define a new GDR GPL working group on developer experience, the DevX. This workgroup will replace the former debugging workgroup. And by broadening the topic, we expect to gather more participants in the group, from several disciplines, in order to have open discussions on the scientific problems related to the developers' needs and possible solutions. (from June 2024) GDR GPL
- Steven Costiou: Program Committee member Prix de thèse du GDR-GPL 2024
- Larisa Safina: Ethics Committee Member Microservices Community

11.1.6 Scientific expertise

- Anne Etien: Evaluation for Agence Nationale de la Recherche ANR
- Anne Etien: Evaluation for the Natural Sciences and Engineering Research Council of Canada NSERC
- Anne Etien: Evaluation for Association Nationale de la Recherche et de la Technologie ANRT

11.1.7 Research administration

- Anne Etien animates the CRIStAL thematic group of Software Engineering.
- Anne Etien: Directrice des Études de la L3 MIAGE, Université de Lille
- Anne Etien: Elected as a Member of CNU (section 27).
- Anne Etien is elected member of the center committee of the Inria Lille center.
- Anne Etien is elected member of the Computer Science Department council of University of Lille
- Anne Etien is member of the Scientific council of CRIStAL
- Marcus Denker was a member of the AGOS board (Section culture) of Inria Lille till 2024

11.2 Teaching - Supervision - Juries

- Master: Stéphane Ducasse, Meta, Université de Lille, 12hTD
- Master: Stéphane Ducasse, Conception avancée, Université de Lille, 60hTD
- Master: Steven Costiou, Advanced Object Oriented Design, IMT Lille Douai, 30hCM
- Master: Steven Costiou, Conception et modélisation objet, Polytech Lille, 10hCM

- Master: Guillermo Polito, Analyse et Verification de Logiciel, Université de Lille, 32h CM
- Master: Guillermo Polito, Conception et Paradigmes de Programmations par la Pratique, Université de Lille, 48hTD
- Licence: Guillermo Polito, Meta, Université de Lille, 13hTD
- Master: Gabriel Darbord, Conception avancée, Université de Lille, 18hTD
- Licence: Gabriel Darbord, JavaScript Full Stack, Université de Lille, 18hTD
- Master: Sebastian Jordan Montaño, Advanced Design, IMT Lille, 38hTD
- 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: Larisa Safina, Métaprogrammation, construction d'interpréteurs, Université de Lille, 24h TD-TP
- Licence: Larisa Safina, Qualité de développement, Université de Lille, 12h TP
- Licence: Larisa Safina, Automatisation de la chaîne de production, Université de Lille, 9h TP
- Licence: Nicolas Anquetil, Maintenance applicative, 18, L3, IUT de Lille
- Licence: Cyril Ferlicot-Delbecque, Génie logiciel, 18hTD, L3, Université de Lille
- Master: Marcus Denker, 2 hours, Advanced Reflection, VUB Brussels, Belgium.

11.2.1 Supervision

- Finished PhD: Iona Thomas, Elements of Language Strenghtening, Stéphane Ducasse, Pablo Tesone, Guillermo Polito. November 7th, 2024, [25]
- Finished PhD: Aless Hosry, Model transformations for automatic source code modification, Nicolas Anquetil. December 6, 2024, [24]
- PhD in progress: Gabriel Darbord, Automatic Test Generation, since october 2022, Inria through the EPC with BL, Anne Etien, Nicolas Anquetil.
- PhD in progress: Valentin Bourcier, Reducing the cost of debugging with the first generation of object-centric debuggers, since october 2022, Inria, Steven Costiou
- PhD in progress: Nahuel Palumbo, Virtual Machine Generation Techniques, since november 2022, Inria, Stéphane Ducasse, Guillermo Polito.
- 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, Imen Sayar 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: Faouzi Rayane, Inria / IMT started oct. 2024, Stéphane Ducasse, L. Fabresse, Pablo Tesone
- PhD in progress: Omar Abedelkader, Inria / Labri / Cirad started oct. 2024, Stéphane Ducasse, R. Robbes, O. Zaitsev

Student interns

- · Ikimath Adouke Adeoye supervised by Clotilde Toullec Improving the Moose importer
- · Quentin Moutte supervised by Stéphane Ducasse Microdown
- · Redwane Engels supervised by Florent Berthaut and Stéphane Ducasse Recoring Scores
- Alexis Cnockaert supervised by Stéphane Ducasse UI for Music
- Sarah Tathy supervised by Stéphane Ducasse and Hernan Federico Morales Contact manager
- · Angel Hely supervised by Cyril Ferlicot-Delbecque Improving quality rules
- · 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

11.2.2 Juries

- Anne Etien: Ali Benjilany, PhD "Une approche pour la modélisation et l'évaluation de l'état d'alignement opérationnel entre processus métier et applicatifs logiciels", 19 décembre 2024, Université de Nantes, as reviewer
- Anne Etien: Matthias Pasquier, PhD "Débogage multivers de spécifications non-déterministes", 19
 décembre 2024, ENSTA Brest, as reviewer
- Anne Etien: Salman Fahrat, PhD "Reconfiguration dynamique sûre des applications comportant des options fonctionnelles", 11 juillet 2024, Université de Lille, as president
- Anne Etien: Corentin Latappy, PhD "Les pratiques de code : de la documentation à la détection", juin 2024, Université de Bordeaux, as reviewer
- Anne Etien: Yassine El Amraoui, PhD "Faciliter l'inclusion humaine dans le processus de science des données : de la capture des exigences métier à la conception d'un workflow d'apprentissage automatique opérationnel", mai 2024, Université Côte d'Azur, as reviewer
- Guillermo Polito: Quentin Ducasse, PhD "Hardware security for just-in-time compilation in language virtual machines", 29 Mars 2024, ENSTA Bretagne, as invited
- Guillermo Polito: Javier Pimas, PhD "Objetos vivos a fondo: removiendo las barreras entre aplicaciones y máquinas virtuales", 12 September 2024, Universidad de Buenos Aires, as reviewer

11.3 Popularization

11.3.1 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
- Sebastian Jordan Montaño organized the PhD GL students day in which the PhD students from different teams (and engineers) gathered to present their research and to exchange

11.3.2 Productions (articles, videos, podcasts, serious games, ...)

- Release of the book *Application building with Spec 2.0*, Koen De Hondt and Stéphane Ducasse with Esteban Lorenzano and Sebastian Jordan Montaño, ISBN 9782322478712 [23]
- Release of the book: *Building a minimal reflective kernel* Stéphane Ducasse, ISBN: 9782322506187 [22]
- The MOOC *Advanced object oriented design and development with Pharo*, originally realeased in 2023, is being run by fun-mooc (From Jun 17, 2024 to Jun 16, 2025). 1031 registerered articipants fun-mooc.fr

11.3.3 Participation in Live events

Domenico Cipriani gave a concert the 17th October at the Maison des Etudiants using Pharo and joined an improvisation session of Muzzix 14/10 using Pharo as his instrument.

11.3.4 Others science outreach relevant activities

• EVREF organizes Public Pharo Sprints every last Friday of the month. Reponsable: Marcus Denker

12 Scientific production

12.1 Major publications

[1] 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-0 3962007.

12.2 Publications of the year

International journals

- [2] V. Bourcier, S. Costiou, M. I. Willembrinck Santander, A. Vanègue and A. Etien. 'Time-traveling object-centric breakpoints'. In: *Journal of Computer Languages* (June 2024), p. 101285. DOI: 10.10 16/j.cola.2024.101285. URL: https://inria.hal.science/hal-04629161 (cit. on p. 17).
- [3] V. Bourcier, M. Willembrinck, A. Vanègue, S. Ducasse, A. Etien and S. Costiou. 'Time-Traveling Queries: Extensible Tools for Faster Program Comprehension.' In: *The Journal of Object Technology* 23.1 (2024), p. 1. DOI: 10.5381/jot.2024.23.1.a7. URL: https://inria.hal.science/hal-0 4841228. In press (cit. on p. 17).

- [4] A. Hosry, V. Aranega and N. Anquetil. 'MoTion: A new declarative object matching approach in Pharo'. In: *Journal of Computer Languages* 81 (4th Sept. 2024), p. 101290. DOI: 10.1016/j.cola.2 024.101290. URL: https://inria.hal.science/hal-04724509 (cit. on p. 19).
- [5] P. Laborde, Y. Le Goff, É. Le Pors, A. Plantec and S. Costiou. 'Live Application Programming in the Defense Industry with the Molecule Component Framework'. In: *Journal of Computer Languages* (June 2024), p. 101286. DOI: 10.1016/j.cola.2024.101286. URL: https://inria.hal.scienc e/hal-04629166 (cit. on p. 20).
- [6] J. P. Sandoval Alcocer, H. Camacho-Jaimes, G. Galindo-Gutierrez, A. Neyem, A. Bergel and S. Ducasse. 'On the Use of Statistical Machine Translation for Suggesting Variable Names for Decompiled Code: The Pharo Case'. In: *Journal of Computer Languages* (28th Apr. 2024). URL: https://inria.hal.s cience/hal-04564690 (cit. on p. 20).
- [7] B. Šarenac, N. Anquetil, S. Ducasse and P. Tesone. 'A New Architecture Reconciling Refactorings and Transformations'. In: *Journal of Computer Languages* (21st July 2024). DOI: 10.1016/j.cola .2024.101273.URL: https://hal.science/hal-04576527 (cit. on pp. 18, 25).
- [8] I. Thomas, S. Ducasse, P. Tesone and G. Polito. 'Pharo: a reflective language -Analyzing the reflective API and its internal dependencies'. In: *Journal of Computer Languages* 80 (2024), p. 101274. DOI: 10.1016/j.cola.2024.101274. URL: https://hal.science/hal-04576440 (cit. on p. 16).

International peer-reviewed conferences

- [9] V. Bourcier, A. Bergel, A. Etien and S. Costiou. 'Debugging Activity Blueprint'. In: The twelfth IEEE Working Conference on Software Visualization (VISSOFT 2024). Pharo. Flagstaff (Arizona), United States, 6th Oct. 2024. URL: https://hal.science/hal-04699796 (cit. on p. 17).
- [10] V. Bourcier and S. Costiou. 'Scopeo: an Object-Centric Debugging Approach for Exploring Object-Oriented Programs'. In: 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). Rovaniemi, Finland, 12th Mar. 2024. DOI: 10.1109/SANER60148.2024.0 0040. URL: https://inria.hal.science/hal-04627606 (cit. on p. 17).
- [11] D. Cipriani, A. Anatrini and S. Jordan Montaño. 'PHAUSTO: EMBEDDING THE FAUST COMPILER IN THE PHARO WORLD'. In: *Proceedings of the 4th International Faust Conference*. 4th International Faust Conference (IFC 2024). Turin, Italy, 21st Nov. 2024. URL: https://hal.science/hal-0483 7510 (cit. on pp. 21, 25).
- [12] D. Cipriani, N. Palumbo, S. J. Montaño and S. Ducasse. 'Phausto: fast and accessible DSP programming for sound and music creation in Pharo'. In: IWST 2024: International Workshop on Smalltalk Technologies. Lille, France, 9th July 2024. URL: https://hal.science/hal-04826894 (cit. on pp. 21, 25).
- [13] G. Darbord, F. Vandewaeter, A. Etien, N. Anquetil and B. Verhaeghe. 'Modest-Pharo: Unit Test Generation for Pharo Based on Traces and Metamodels'. In: IWST 2024: International Workshop on Smalltalk Technologies. Lille, France, 9th July 2024. URL: https://hal.science/hal-04622256 (cit. on p. 18).
- [14] P. Laborde, É. Le Pors, Y. Le Goff, A. Plantec and S. Costiou. 'Ergonomic evaluations of Human-Machine Interfaces in the defense business: an example of a collaborative Maritime Surveillance system'. In: IHM'24 - 35e Conférence Internationale Francophone sur l'Interaction Humain-Machine. Vol. IHM'24 : Actes étendus de la 35ème conférence Francophone sur l'Interaction Humain-Machine. Paris, France, 25th Mar. 2024. URL: https://hal.science/hal-04498410 (cit. on p. 20).
- [15] S. J. Montaño, G. Polito, S. Ducasse and P. Tesone. 'Evaluating Finalization-Based Object Lifetime Profiling'. In: International Symposium on Memory Management. Copenhagen, Denmark, 2024. DOI: 10.1145/3652024.3665514. URL: https://hal.science/hal-04581342 (cit. on p. 19).
- [16] S. J. Montaño, J. P. Sandoval Alcocer, G. Polito, S. Ducasse and P. Tesone. 'MethodProxies: A Safe and Fast Message-Passing Control Library'. In: IWST 2024: International Workshop on Smalltalk Technologies, July 8-11, 2024, Lille, France. Lille, France, 8th July 2024. URL: https://hal.scienc e/hal-04708729 (cit. on p. 19).

- [17] T. Nguyen, S. Ebersold, I. Sayar and J.-M. Bruel. 'Identifying and fixing ambiguities in, and semantically accurate formalisation of, behavioural requirements'. In: *Software and Systems Modeling*. 27 th International Conference on Model Driven Engineering Languages and Systems (MODELS 2024). Linz, Austria, Sept. 2024, à paraître. DOI: 10.1007/s10270-023-01142-0. URL: https://hal.sc ience/hal-04700472 (cit. on p. 15).
- [18] B. Šarenac, S. Ducasse, G. Polito and G. Rakic. 'Modular and Extensible Extract Method'. In: International Workshop on Smalltalk Technologies - IWST 24. Lille, France, 9th July 2024. URL: https: //inria.hal.science/hal-04670318 (cit. on pp. 18, 25).
- [19] I. Thomas, S. Ducasse, G. Polito and P. Tesone. 'Assessing Reflection Usage with Mutation Testing Augmented Analysis'. In: 21st International Conference on Software and Systems Reuse (ICSR 2024). Limassol, Cyprus, 19th June 2024. URL: https://inria.hal.science/hal-04600101 (cit. on p. 16).

Conferences without proceedings

- [20] 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://ha l.science/hal-04841805 (cit. on p. 16).
- [21] 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.

Scientific books

- [22] S. Ducasse. Building a minimal reflective kernel. 1st Dec. 2024. URL: https://hal.science/hal-04888214 (cit. on p. 33).
- [23] K. de Hondt, S. Ducasse, S. Jordan Montaño and E. Lorenzano. Application Building with Spec 2.0. 2024. URL: https://hal.science/hal-04888256 (cit. on p. 33).

Doctoral dissertations and habilitation theses

- [24] A. Hosry. 'External Dependencies in Programs: Specification, Detection and Incorrectness.' Lille, 6th Dec. 2024. URL: https://hal.science/tel-04847289 (cit. on p. 31).
- [25] I. Thomas. 'Reflection Analysis and First Steps in Its Control'. Université de Lille, 7th Nov. 2024. URL: https://inria.hal.science/tel-04845616 (cit. on p. 31).

Reports & preprints

- [26] V. Bourcier, P. Rani, M. I. Willembrinck Santander, A. Bacchelli and S. Costiou. *Empirically Evaluating the Impact of Object-Centric Breakpoints on the Debugging of Object-Oriented Programs*. 11th Feb. 2025. DOI: 10.1145/3715759. URL: https://hal.science/hal-04948470.
- [27] M. Denker, N. Anquetil, C. Bortolaso, S. Costiou, S. Ducasse, A. Etien, G. Polito, L. Safina and I. Sayar. 2023 ACTIVITY REPORT Project-Team EVREF. INRIA, 1st Apr. 2024. URL: https://hal.science/hal-04556508.
- [28] A. Etien and N. Anquetil. Automatic Recommendations for Evolving Relational Databases Schema. Université de Lille, Faculté de Sciences et Technologies, 12th Apr. 2024. URL: https://inria.hal .science/hal-04546264 (cit. on p. 15).

Software

- [29] [SW] V. Bourcier and S. Costiou, Scopeo version 1.0.0, 14th Mar. 2024. INRIA Lille Nord Europe. LIC: MIT License. HAL: (hal-04880231), URL: https://hal.science/hal-04880231, VCS: https://github.com/scopeo-project/ScopeoExampleERA, SWHID: (swh:1:dir:b006859e 0291f62b4e6868f51f8c705e87fb4e3d;origin=https://github.com/scopeo-project/Sc opeoExampleERA;visit=swh:1:snp:1ce6fb9de2f9a812dcda8eb1e30a3a068d2782a9;anch or=swh:1:rev:eb1057b63339f394bdd712ad3a649c9492157f1f).
- [30] [SW] S. Costiou, Phex version 1.0, 29th Dec. 2024. Centre Inria de l'Université de Lille. LIC: MIT License. HAL: (hal-04858390), URL: https://inria.hal.science/hal-04858390, VCS: https://github.com/Pharo-XP-Tools/Phex,SWHID: (swh:1:dir:fccf734ffb8d01e9c3b0 3454422e97be18770f6a;origin=https://github.com/Pharo-XP-Tools/Phex;visit=swh :1:snp:cf48b926d985066e803a7c3c822eaf0b8c8bbf3c;anchor=swh:1:rev:033c1e523df ea5a12fffee1b8d88120f58e68982).
- [31] [SW] S. Costiou and A. Vanègue, Chest version 0.5.1, 29th Dec. 2024. Centre Inria de l'Université de Lille. LIC: MIT License. HAL: (hal-04858337), URL: https://inria.hal.science/hal-04858 337, VCS: https://github.com/pharo-spec/Chest, SWHID: (swh:1:dir:e17ddf4f775546a6 e5b4548879eee3e07e6227b3; origin=https://github.com/pharo-spec/Chest; visit=sw h:1:snp:4f017094ae09aef0243db4b693cecac91e637b1a; anchor=swh:1:rev:feff10b5c8 bc2888d95cc42dc189c9688dfefc52).
- [32] [SW] S. Costiou and A. Vanègue, *Debugging Spy* version 1.0, 29th Dec. 2024. Centre Inria de l'Université de Lille. LIC: MIT License. HAL: (hal-04858378), URL: https://inria.hal.sci ence/hal-04858378, VCS: https://github.com/Pharo-XP-Tools/DebuggingSpy, SWHID: (swh:1:dir:0f63d67301c0ad3174d17c89a13b52e595837877;origin=https://github.co m/Pharo-XP-Tools/DebuggingSpy;visit=swh:1:snp:ae7703ee9ee6f77eb10697b7d9fdf9 0678a768a0;anchor=swh:1:rev:fbec9a14cbdaa478f498196c0f993062441ef3ae).
- [33] [SW] S. Costiou and A. Vanègue, Sindarin version 12, 26th Apr. 2024. Centre Inria de l'Université de Lille. LIC: MIT License. HAL: (hal-04850901), URL: https://inria.hal.science/hal-048 50901, SWHID: (swh:1:dir:4a7c622c61fcb3be319c91e5871b8c2d982b02c9;origin=https: //github.com/pharo-spec/ScriptableDebugger;visit=swh:1:snp:5ada4e8d86339556b d6e8fe7b3026173d012c962;anchor=swh:1:rev:368543cd0b13bffbd1100db94a11ecc3bb1 cda60).
- [34] [SW] N. Hartl and M. Denker, *Soil* version v2, 28th Aug. 2024. ApptiveGrid; INRIA. LIC: MIT License. HAL: (hal-04726251), URL: https://hal.science/hal-04726251, VCS: https://github.co m/ApptiveGrid/Soil, SWHID: (swh:1:dir:7ab481741958af9189981e1c557aacfad5d78d7c ;origin=https://github.com/ApptiveGrid/Soil;visit=swh:1:snp:35d065c034bfd37a 5f0fe9a16e23b593f547de8d;anchor=swh:1:rev:0541435382436034a584a3aec8e1acada4 8c8bd9).