IN PARTNERSHIP WITH:
CNRS

Université des sciences et technologies de Lille (Lille 1)

Activity Report 2012

Project-Team RMOD

Analyses and Languages Constructs for Object-Oriented Application Evolution

IN COLLABORATION WITH: Laboratoire d'informatique fondamentale de Lille (LIFL)

RESEARCH CENTER
Lille - Nord Europe

THEME
Distributed Systems and Services

Table of contents

Project-Team RMOD

Keywords: Software Engineering, Software Evolution, Maintenance, Reflective Programming Languages, Dynamic Languages

Beginning of the Team: 2009-07-01, End of the Team: 2013-12-31.

1. Members

Research Scientists
- Stéphane Ducasse [Team leader, Senior Researcher (DR2), HdR]
- Marcus Denker [Researcher (CR1)]

Faculty Members
- Nicolas Anquetil [Associate Professor (MCF) USTL – IUT]
- Damien Cassou [Associate Professor (MCF) USTL – University Lille 1, joined in September]
- Anne Etien [Associate Professor (MCF), USTL – Polytech'Lille, joined in November]
- Damien Pollet [Associate Professor (MCF) USTL – Telecom Lille 1]

Engineers
- Clement Bera [Engineer]
- Christophe Demarey [Permanent Engineer]
- Guillaume Larcheveque [Engineer]
- Esteban Lorenzano [Expert Engineer]
- Olivier Auverlot [Engineer, USTL – LIFL (20%)]
- Nicolas Petton [Engineer]
- Igor Stasenko [Expert Engineer]

PhD Students
- Jean-Baptiste Arnaud
- Camillo Bruni
- Martin Dias [Inria Region, joined in December]
- Andre Hora
- Mariano Martinez-Peck [Ecole des Mines de Douai, co-supervision]
- Nick Papoylias [Ecole des Mines de Douai, co-supervision]
- Guillermo Polito [Ecole des Mines de Douai, co-supervision]
- Camille Teruel [Inria – DGA]
- Veronica Uquillas-Gomez [Vrije Universiteit Brussel, co-tutelle]

Post-Doctoral Fellows
- Muhammad Bhatti
- Simon Allier

Administrative Assistant
- Christelle Gasperini [Secretary (SAR)]

Others
- Benjamin Arezki [Student]
- Fernando Olivero [Visiting Student]
- Benjamin van Ryseghem [Intern, USTL – University Lille 1]
- Erwan Douaille [Intern, USTL – University Lille 1]
- Ezequiel La mónica [Intern, Universidad de Buenos Aires, Argentina]

2. Overall Objectives

2.1. Introduction

Keywords: Software evolution, Maintenance, Program visualization, Program analyses, Meta modelling, Software metrics, Quality models, Object-oriented programming, Reflective programming, Traits, Dynamically typed languages, Smalltalk.

RMoD's general vision is defined in two objectives: remodularization and modularity constructs. These two views are the two faces of a same coin: maintenance could be eased with better engineering and analysis tools and programming language constructs could let programmers define more modular applications.

2.2. Reengineering and remodularization

While applications must evolve to meet new requirements, few approaches analyze the implications of their original structure (modules, packages, classes) and their transformation to support their evolution. Our research will focus on the *remodularization* of object-oriented applications. Automated approaches including clustering algorithms are not satisfactory because they often ignore user inputs. Our vision is that we need better approaches to support the transformation of existing software. The reengineering challenge tackled by RMoD is formulated as follows:

How to help remodularize existing software applications?

We are developing analyses and algorithms to remodularize object-oriented applications. This is why we started studying and building tools to support the *understanding of applications* at the level of packages and modules. This allows us to understand the results of the *analyses* that we are building.

2.3. Constructs for modular and isolating programming languages

Dynamically-typed programming languages such as JavaScript are getting new attention as illustrated by the large investment of Google in the development of the Chrome V8 JavaScript engine and the development of a new dynamic language DART. This new trend is correlated to the increased adoption of dynamic programming languages for web-application development, as illustrated by Ruby on Rails, PHP and JavaScript. With web applications, users expect applications to be always available and getting updated on the fly. This continuous evolution of application is a real challenge [46]. Hot software evolution often requires *reflective* behavior and features. For instance in CLOS and Smalltalk each class modification automatically migrates existing instances on the fly.

At the same time, there is a need for *software isolation i.e.,*, applications should reliably run co-located with other applications in the same virtual machine with neither confidential information leaks nor vulnerabilities. Indeed, often for economical reasons, web servers run multiple applications on the same virtual machine. Users need confined applications. It is important that (1) an application does not access information of other applications running on the same virtual machine and (2) an application authorized to manipulate data cannot pass such authorization or information to other parts of the application that should not get access to it.

Static analysis tools have always been confronted to reflection [28], [43]. Without a full treatment of reflection, static analysis tools are both incomplete and unsound. Incomplete because some parts of the program may not be included in the application call graph, and unsound because the static analysis does not take into account reflective features [52]. In reflective languages such as F-Script, Ruby, Python, Lua, JavaScript, Smalltalk and Java (to a certain extent), it is possible to nearly change any aspect of an application: change objects, change classes dynamically, migrate instances, and even load untrusted code.

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

What are the language modularity constructs to support isolation?

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

2.4. Highlights of the Year

- *Emergence Award*: Synectique is a startup project of RMoD around building customized software analysis tools. The project participated in the competition by French Ministry of research and higher education for innovative projects ("Concours OSEO"). The project was selected in the competition and won an award of 30K€ to develop its activities (http://rmod.lille.inria.fr/web/pier/blog/synectique-oseo).
- Moose 4.6 (our open-source reengineering platform) was released (http://www.moosetechnology.org/).
- Pharo 1.4 (our open-source language and environment) was released (http://www.pharo-project.org).
- RMoD organized the first Pharo Conference during two days in May (60 participants).
- RMoD participated to the organization of the ESUG conference in Ghent, Belgium in August (130 participants).
- Marcus Denker got promoted to CR1.
- RMoD launched the Pharo Consortium and the Pharo Association.

3. Scientific Foundations

3.1. Software Reengineering

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

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

[1] Traits are groups of methods that can be composed orthogonally to simple inheritance. Contrary to mixin, the class has the control of the composition and conflict management.

To help us answer these questions, we work on enriching Moose, our reengineering environment, with a new set of analyses [37], [36]. We decompose our approach in three main and potentially overlapping steps:

1. Tools for understanding applications,
2. Remodularization analyses,
3. Software Quality.

3.1.1. Tools for understanding applications

Context and Problems. We are studying the problems raised by the understanding of applications at a larger level of granularity such as packages or modules. We want develop a set of conceptual tools to support this understanding.

Some approaches based on Formal Concept Analysis (FCA) [65] show that such an analysis can be used to identify modules. However the presented examples are too small and not representative of real code.

Research Agenda.

FCA provides an important approach in software reengineering for software understanding, design anomalies detection and correction, but it suffers from two problems: (i) it produces lattices that must be interpreted by the user according to his/her understanding of the technique and different elements of the graph; and, (ii) the lattice can rapidly become so big that one is overwhelmed by the mass of information and possibilities [23]. We look for solutions to help people putting FCA to real use.

3.1.2. Remodularization analyses

Context and Problems. It is a well-known practice to layer applications with bottom layers being more stable than top layers [53]. Until now, few works have attempted to identify layers in practice: Mudpie [67] is a first cut at identifying cycles between packages as well as package groups potentially representing layers. DSM (dependency structure matrix) [66], [61] seems to be adapted for such a task but there is no serious empirical experience that validates this claim. From the side of remodularization algorithms, many were defined for procedural languages [49]. However, object-oriented programming languages bring some specific problems linked with late-binding and the fact that a package does not have to be systematically cohesive since it can be an extension of another one [68], [40].

As we are designing and evaluating algorithms and analyses to remodularize applications, we also need a way to understand and assess the results we are obtaining.

Research Agenda. We work on the following items:

Layer identification. We propose an approach to identify layers based on a semi-automatic classification of package and class interrelationships that they contain. However, taking into account the wish or knowledge of the designer or maintainer should be supported.

Cohesion Metric Assessment. We are building a validation framework for cohesion/coupling metrics to determine whether they actually measure what they promise to. We are also compiling a number of traditional metrics for cohesion and coupling quality metrics to evaluate their relevance in a software quality setting.

3.1.3. Software Quality

Research Agenda. Since software quality is fuzzy by definition and a lot of parameters should be taken into account we consider that defining precisely a unique notion of software quality is definitively a Grail in the realm of software engineering. The question is still relevant and important. We work on the two following items:

Quality models. We studied existing quality models and the different options to combine indicators — often, software quality models happily combine metrics, but at the price of losing the explicit relationships between the indicator contributions. There is a need to combine the results of one metric over all the software components of a system, and there is also the need to combine different metric results for any software component. Different combination methods are possible that can give very different results. It is therefore important to understand the characteristics of each method.

Bug prevention. Another aspect of software quality is validating or monitoring the source code to avoid the apparition of well known sources of errors and bugs. We work on how to best identify such common errors, by trying to identify earlier markers of possible errors, or by helping identifying common errors that programmers did in the past.

3.2. Language Constructs for Modular Design

While the previous axis focuses on how to help remodularizing existing software, this second research axis aims at providing new language constructs to build more flexible and recomposable software. We will build on our work on traits [63], [38] and classboxes [24] but also start to work on new areas such as isolation in dynamic languages. We will work on the following points: (1) Traits and (2) Modularization as a support for isolation.

3.2.1. Traits-based program reuse

Context and Problems. Inheritance is well-known and accepted as a mechanism for reuse in object-oriented languages. Unfortunately, due to the coarse granularity of inheritance, it may be difficult to decompose an application into an optimal class hierarchy that maximizes software reuse. Existing schemes based on single inheritance, multiple inheritance, or mixins, all pose numerous problems for reuse.

To overcome these problems, we designed a new composition mechanism called Traits [63], [38]. Traits are pure units of behavior that can be composed to form classes or other traits. The trait composition mechanism is an alternative to multiple or mixin inheritance in which the composer has full control over the trait composition. The result enables more reuse than single inheritance without introducing the drawbacks of multiple or mixin inheritance. Several extensions of the model have been proposed [35], [57], [25], [39] and several type systems were defined [41], [64], [58], [51].

Traits are reusable building blocks that can be explicitly composed to share methods across unrelated class hierarchies. In their original form, traits do not contain state and cannot express visibility control for methods. Two extensions, stateful traits and freezable traits, have been proposed to overcome these limitations. However, these extensions are complex both to use for software developers and to implement for language designers.

Research Agenda: Towards a pure trait language. We plan distinct actions: (1) a large application of traits, (2) assessment of the existing trait models and (3) bootstrapping a pure trait language.

- To evaluate the expressiveness of traits, some hierarchies were refactored, showing code reuse [27]. However, such large refactorings, while valuable, may not exhibit all possible composition problems, since the hierarchies were previously expressed using single inheritance and following certain patterns. We want to redesign from scratch the collection library of Smalltalk (or part of it). Such a redesign should on the one hand demonstrate the added value of traits on a real large and redesigned library and on the other hand foster new ideas for the bootstrapping of a pure trait-based language.

  In particular we want to reconsider the different models proposed (stateless [38], stateful [26], and freezable [39]) and their operators. We will compare these models by (1) implementing a trait-based collection hierarchy, (2) analyzing several existing applications that exhibit the need for traits. Traits may be flattened [56]. This is a fundamental property that confers to traits their simplicity and expressiveness over Eiffel's multiple inheritance. Keeping these aspects is one of our priority in forthcoming enhancements of traits.

- Alternative trait models. This work revisits the problem of adding state and visibility control to traits. Rather than extending the original trait model with additional operations, we use a fundamentally different approach by allowing traits to be lexically nested within other modules. This enables traits to express (shared) state and visibility control by hiding variables or methods in their lexical scope. Although the traits' "flattening property" no longer holds when they can be lexically nested, the combination of traits with lexical nesting results in a simple and more expressive trait model. We formally specify the operational semantics of this combination. Lexically nested traits are fully

implemented in AmbientTalk, where they are used among others in the development of a Morphic-like UI framework.

- We want to evaluate how inheritance can be replaced by traits to form a new object model. For this purpose we will design a minimal reflective kernel, inspired first from ObjVlisp [33] then from Smalltalk [44].

3.2.2. Reconciling Dynamic Languages and Isolation

Context and Problems. More and more applications require dynamic behavior such as modification of their own execution (often implemented using reflective features [48]). For example, F-script allows one to script Cocoa Mac-OS X applications and Lua is used in Adobe Photoshop. Now in addition more and more applications are updated on the fly, potentially loading untrusted or broken code, which may be problematic for the system if the application is not properly isolated. Bytecode checking and static code analysis are used to enable isolation, but such approaches do not really work in presence of dynamic languages and reflective features. Therefore there is a tension between the need for flexibility and isolation.

Research Agenda: Isolation in dynamic and reflective languages. To solve this tension, we will work on *Sure*, a language where isolation is provided by construction: as an example, if the language does not offer field access and its reflective facilities are controlled, then the possibility to access and modify private data is controlled. In this context, layering and modularizing the meta-level [29], as well as controlling the access to reflective features [30], [31] are important challenges. We plan to:

- Study the isolation abstractions available in erights (http://www.erights.org) [55], [54], and Java's class loader strategies [50], [45].
- Categorize the different reflective features of languages such as CLOS [47], Python and Smalltalk [59] and identify suitable isolation mechanisms and infrastructure [42].
- Assess different isolation models (access rights, capabilities [60]...) and identify the ones adapted to our context as well as different access and right propagation.
- Define a language based on
  - the decomposition and restructuring of the reflective features [29],
  - the use encapsulation policies as a basis to restrict the interfaces of the controlled objects [62],
  - the definition of method modifiers to support controlling encapsulation in the context of dynamic languages.

An open question is whether, instead of providing restricted interfaces, we could use traits to grant additional behavior to specific instances: without trait application, the instances would only exhibit default public behavior, but with additional traits applied, the instances would get extra behavior. We will develop *Sure*, a modular extension of the reflective kernel of Smalltalk (since it is one of the languages offering the largest set of reflective features such as pointer swapping, class changing, class definition...) [59].

4. Software

4.1. Moose

Participants: Stéphane Ducasse [correspondant], Muhammad Bhatti, Andre Hora, Nicolas Anquetil, Tudor Gîrba [University of Bern].

Web: http://www.moosetechnology.org/

The platform. 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, a model repository, and generic GUI support for querying, browsing and grouping. The development of Moose began at the Software Composition Group in 1997, and is currently contributed to and used by researchers in at least seven European universities. Moose offers an extensible meta-described metamodel, a query engine, a metric engine and several visualizations. Moose is currently in its fourth major release and comprises 55,000 lines of code in 700 classes.

The RMoD team is currently the main maintainer of the Moose platform. There are 200 publications (journal, international conferences, PhD theses) based on execution or use of the Moose environment.

The first version running on top of Pharo (Moose 4.0) was released in June 2010. In February 2012, Moose 4.6 was released.
Here is the self-assessment of the team effort following the grid given at http://www.inria.fr/institut/organisation/instances/commission-d-evaluation.

- **(A5)** Audience : 5 – Moose is used by several research groups, a consulting company, and some companies using it in ad-hoc ways.
- **(SO4)** Software originality : 4 – Moose aggregates the last results of several research groups.
- **(SM4)** Software Maturity : 4 – Moose is developed since 1996 and got two main redesign phases.
- **(EM4)** Evolution and Maintenance : 4 – Moose will be used as a foundation of our Synectique start up so its maintenance is planned.
- **(SDL4)** Software Distribution and Licensing : 4 – Moose is licensed under BSD
- **(OC)** Own Contribution : (Design/Architecture)DA-4, (Coding/Debugging)-4, (Maintenance/Support)-4, (Team/Project Management)-4

4.2. Pharo

Participants: Marcus Denker [correspondant], Damien Cassou, Stéphane Ducasse, Esteban Lorenzano, Mariano Martinez-Peck, Damien Pollet, Igor Stasenko, Veronica Uquillas-Gomez.

Web: http://www.pharo-project.org/

The platform. Pharo is a new open-source Smalltalk-inspired language and environment. 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 Smalltalk applications.

The first stable version, Pharo 1.0, was released in 2010. The development of Pharo accelerated in 2011 and 2012: Versions 1.2 to 1.4 have been released (with more than 2400 closed issues), and the development branch (2.0) has seen already over 398 incremental releases as of mid November 2012. In 2012, RMoD organized the first *Pharo Conference* during two days in May with 60 participants.

Additionally, in November 2012 RMoD launched the Pharo Consortium (http://www.pharo-project.org/community/consortium) and the Pharo Association (http://association.pharo.org/). 25 companies already shown interest in supporting the consortium.

RMoD is the main maintainer and coordinator of Pharo.

Here is the self-assessment of the team effort following the grid given at http://www.inria.fr/institut/organisation/instances/commission-d-evaluation.

- **(A5)** Audience: 5 – Used in many universities for teaching, more than 25 companies.
- **(SO3)** Software originality : 3 – Pharo offers a classical basis for some aspects (UI). It includes new frameworks and concepts compared to other Smalltalk implementations.
- **(SM4)** Software Maturity: 4 – Bug tracker, continuous integration, large test suites are on place.
- **(EM4)** Evolution and Maintenance: 4 – Active user group, consortium and association had just been set up.
- **(SDL4)** Software Distribution and Licensing: 4 – Pharo is licensed under MIT.
- **(OC5)** Own Contribution: (Design/Architecture) DA-5, (Coding/Debugging) CD-5, (Maintenance/Support) MS-5, (Team/Project Management) TPM-5

4.3. Fuel

Participants: Martin Dias [Correspondant], Mariano Martinez-Peck.

Web: http://rmod.lille.inria.fr/web/pier/software/fuel

Objects in a running environment are constantly being born, mutating their status and dying in the volatile memory of the system. The goal of serializers is to store and load objects either in the original environment or in another one. Fuel is a general-purpose serializer based on four principles: (1) speed, through a compact binary format and a pickling algorithm which obtains the best performance on materialization; (2) good object-oriented design, without any special help from the virtual machine; (3) specialized for Pharo, so that core objects (such as contexts, block closures and classes) can be serialized too; (4) flexible about how to serialize each object, so that objects are serialized differently depending on the context.
Here is the self-assessment of the team effort following the grid given at http://www.inria.fr/institut/organisation/instances/commission-d-evaluation.

- **(A4)** Audience: 4 – Large audience software, usable by people inside and outside the field with a clear and strong dissemination, validation, and support action plan.
- **(SO3)** Software originality : 3.
- **(SM4)** Software Maturity: 4 – Bug tracker, continuous integration, large test suites are on place.
- **(EM4)** Evolution and Maintenance: 4.
- **(SDL4)** Software Distribution and Licensing: 4 – Fuel is licensed under MIT.
- **(OC5)** Own Contribution: (Design/Architecture) DA-5, (Coding/Debugging) CD-5, (Maintenance/Support) MS-5, (Team/Project Management) TPM-5

4.4. Athens

Participant: Igor Stasenko [Correspondant].

Athens is a vector graphics framework for Pharo.

4.5. Citezen

Participants: Damien Pollet [Correspondant], Stéphane Ducasse.

Web: http://people.untyped.org/damien.pollet/software/citezen/

Citezen is a suite of tools for parsing, validating, sorting and displaying BibTeX databases. This tool suite is integrated within the Pier Content Management System (CMS) and both are implemented on top of Pharo. Citezen aims at replacing and extending BibTeX, in Smalltalk; ideally, features would be similar to BibTeX, CrossTeX, and CSL.

4.6. Handles

Participant: Jean-Baptiste Arnaud [Correspondant].

Web: http://jeanbaptiste-arnaud.eu/handles/

An Handle is a first-class reference to a target object. Handles can alter the behavior and isolate the state of the target object. Handles provide infrastructure to automatically create and wrap new handles when required. A real-time control of handles is possible using a special object called metaHandle.

4.7. Hazelnut

Participants: Guillermo Polito [Correspondant], Benjamin van Ryseghem, Nicolas Paez, Igor Stasenko.

Web: http://rmod.lille.inria.fr/web/pier/software/Seed

Traditionally, Smalltalk-based systems are not bootstrapped because of their ability to evolve by self-modification. Nevertheless, the absence of a bootstrap process exposes many problems in these systems, such as the lack of reproducibility and the impossibility to reach certain evolution paths. Hazelnut is a tool that aims to introduce a bootstrap process into these systems, in particular Pharo.

4.8. Jet

Participant: Veronica Uquillas-Gomez [Correspondant].

Jet is a tool to analyze streams of changes. Jet identifies dependencies between changes and sets of changes and supports cherry picking. Moreover, Jet classifies sets of changes based on their dependencies as a way to ease the analysis of changes within the stream and guide system integrators.

4.9. LegacyParsers

Participants: Muhammad Bhatti [Correspondant], Nicolas Anquetil, Guillaume Larcheveque, Esteban Lorenzano, Gogui Ndong.

As part of our research on legacy software and also for the Synectique company), we started to define several parsers for old languages like Cobol for example. This work is important to help us validate our meta-model and tools against a larger range of existing technologies and to discover the limits of our approach. From our initial results, and the in-depth understanding that it gave us, we are formulating new research objectives in meta-model driven reverse engineering. This work is also important for the spin-off company, as being able to work with such technologies is fundamental.

4.10. Mate

Participants: Marcus Denker [Correspondant], Clement Bera, Camillo Bruni.

Mate is the future research-oriented virtual machine for Pharo. Its goal is to serve as a prototype for researchers to experiment with. As a result, the design of Mate is very simple to understand. As of today, Mate consists of an AST interpreter, a new object memory layout, and a simple garbage collector.

4.11. NativeBoost

Participant: Igor Stasenko [Correspondant].

Web: http://code.google.com/p/nativeboost/

NativeBoost is a Smalltalk framework for generating and running machine code from the language side of Pharo. As part of it comes a foreign function interface that enables calling external C functions from Smalltalk code with minimal effort.

4.12. Nabujito

Participants: Camillo Bruni [Correspondant], Marcus Denker.

Nabujito is a new Just In Time compiler implemented as a Smalltalk application, based on NativeBoost, that does not require changes in the virtual machine.

4.13. Nautilus

Participants: Benjamin Van Ryseghem [Correspondant], Stéphane Ducasse, Igor Stasenko, Camillo Bruni, Esteban Lorenzano.

Nautilus is a new source code browser based on the latest infrastructure representations. Its goal is mainly to replace the current system browser that was implemented in the 80s and that doesn't provide optimal tools for the system as it has evolved.

4.14. SourceCity

Participants: Erwan Douaille [Correspondant], Igor Stasenko, Guillaume Larcheveque, Stéphane Ducasse.

Modern systems are too complex. Understanding and analyzing these systems is very hard and tedious (thousand of classes, millions of lines of code). One needs an overview of the system that allows to discover important parts in the system, weak points, suspicious components. SourceCity is a powerful 3D tool that can help to understand quickly how a system works by taking the metaphor of a city buildings. By looking at tall, large, low building, one can identify different properties of the software components being represented.

4.15. Spec

Participants: Benjamin Van Ryseghem [Correspondant], Stéphane Ducasse, Johan Fabry.

Spec is a programming framework for generating graphical user interfaces inspired by VisualWorks' Subcanvas. The goal of Spec is to tackle the lack of reuse experienced in existing tools. Spec serves as a pluggable layer on top of multiple lower-level graphical frameworks. Many improvements have been noticed in Pharo after the introduction of Spec in terms of speed or number of lines of code while we re-implemented existing tools using Spec.

4.16. VerveineJ

Participants: Nicolas Anquetil [Correspondant], Andre Hora, Guillaume Larcheveque.

Web: Inria project https://gforge.inria.fr/projects/verveinej/.

VerveineJ is a tool to export Java projects into the MSE format, which can then be imported inside Moose (see above). Although VerveineJ is not a research project in itself, it is an important building block for our research in that it allows us to run the Moose platform on legacy Java projects. Another similar tool, Infusion, already existed to fulfil the same needs, but it was closed sources and presented some errors that tainted the results we could obtain.

5. New Results

5.1. Object serializer

Participants: Martin Dias [Correspondant], Mariano Martinez-Peck, Stéphane Ducasse.

Fuel: A Fast General Purpose Object Graph Serializer Since objects need to be stored and reloaded on different environments, serializing object graphs is a very important activity. There is a plethora of serialization frameworks with different requirements and design trade-offs. Most of them are based on recursive parsing of the object graphs, an approach which often is too slow. In addition, most of them prioritize a language-agnostic format instead of speed and language-specific object serialization. For the same reason, such serializers usually do not support features like class-shape changes, global references or executing pre and post load actions. Looking for speed, some frameworks are partially implemented at Virtual Machine (VM) level, hampering code portability and making them difficult to understand, maintain and extend. That is why we work on Fuel, a general-purpose object serializer based on these principles: (1) speed, through a compact binary format and a pickling algorithm which invests time in serialization for obtaining the best performance on materialization; (2) good object-oriented design, without special help at VM; (3) serialize any object, thus have a full-featured language-specific format. We implement and validate this approach in Pharo, where we demonstrate that Fuel is faster than other serializers, even those with special VM support. The extensibility of Fuel made possible to successfully serialize various objects: classes in Newspeak, debugger stacks, and full CMS object graphs [11].

5.2. Cycles and dependencies

Participants: Stéphane Ducasse [Correspondant], Nicolas Anquetil, Muhammad Bhatti.

OZONE: Layer Identification in the presence of Cyclic Dependencies A layered software architecture helps understanding the role of software entities (e.g., packages or classes) in a system and hence, the impact of changes on these entities. However, the computation of an optimal layered organization in the presence of cyclic dependencies is difficult. We develop an approach that (i) provides a strategy supporting the automated detection of cyclic dependencies, (ii) proposes heuristics to break cyclic dependencies, and (iii) computes an organization of software entities in multiple layers even in presence of cyclic dependencies. Our approach performs better than the other existing approaches in terms of accuracy and interactivity, it supports human inputs and constraints. We compare this approach to existing solutions and apply it on two large software systems to identify package layers. The results are manually validated by software engineers of the two systems [12].

Efficient Retrieval and Ranking of Undesired Package Cycles in Large Software Systems Many design guidelines state that a software system architecture should avoid cycles between its packages. Yet such cycles appear again and again in many programs. We believe that the existing approaches for cycle detection are too coarse to assist developers to remove cycles from their programs. We design an efficient algorithm that performs a fine-grained analysis of cycles among application packages. In addition, we define multiple metrics to rank cycles by their level of undesirability, prioritizing cycles that are the more undesired by developers. We compare these multiple ranking metrics on four large and mature software systems in Java and Smalltalk [14].

Resolving cyclic dependencies between packages with Enriched Dependency Structural Matrix Dependency Structural Matrix (DSM) is an approach originally developed for process optimization. It has been successfully applied to identify software dependencies among packages and subsystems. A number of algorithms have been proposed to compute the matrix so that it highlights patterns and problematic dependencies between subsystems. However, existing DSM implementations often miss important information to fully support reengineering effort. For example, they do not clearly qualify and quantify problematic relationships, information that is crucial to support remediation tasks. We propose Enriched Dependency Structural Matrix (eDSM), which provides small multiple views and micro-macro readings by adding fine-grained information in each cell of the matrix. Each cell is enriched with contextual information about (i) the type of dependencies (inheritance, class reference, etc.), (ii) the proportion of referencing entities, (iii) the proportion of referenced entities. We distinguish independent cycles and stress potentially simple fixes for cycles using coloring information. This work is language independent and has been implemented on top of the Moose software analysis platform.We improved the cell content information view based on user feedback and performed multiple validations: two different case studies on Moose and Seaside software; one user study for validating eDSM as a usable approach for developers. Solutions to problems identified with eDSM have been performed and retrofitted in analyzed software [13].

5.3. Warnings and bugs

Participants: Simon Allier [Correspondant], Andre Hora, Nicolas Anquetil, Muhammad Bhatti, Stéphane Ducasse.

A Framework to Compare Alert Ranking Algorithms To improve software quality, rule checkers statically check if a software contains violations of good programming practices. On a real sized system, the alerts (rule violations detected by the tool) may be numbered by the thousands. Unfortunately, these tools generate a high proportion of "false alerts", which in the context of a specific software, should not be fixed. Huge numbers of false alerts may render impossible the finding and correction of "true alerts" and dissuade developers from using these tools. In order to overcome this problem, the literature provides different ranking methods that aim at computing the probability of an alert being a "true one". We propose a framework for comparing these ranking algorithms and identify the best approach to rank alerts. We have selected six algorithms described in literature. For comparison, we use a benchmark covering two programming languages (Java and Smalltalk) and three rule checkers (FindBug, PMD, SmallLint). Results show that the best ranking methods are based on the history of past alerts and their location. We could not identify any significant advantage in using statistical tools such as linear regression or Bayesian networks or ad-hoc methods [15].

Uncovering Causal Relationships between Software Metrics and Bugs Bug prediction is an important challenge for software engineering research that consists in looking for possible early indicators of the presence of bugs in a software. However, despite the relevance of the issue, most experiments designed to evaluate bug prediction only investigate whether there is a linear relation between the predictor and the presence of bugs. However, it is well known that standard regression models can not filter out spurious relations. We describe an experiment to discover more robust evidences towards causality between software metrics (as predictors) and the occurrence of bugs. For this purpose, we have relied on Granger Causality Test to evaluate whether past changes in a given time series are useful to forecast changes in another series. As its name suggests, Granger Test is a better indication of causality between two variables. We present and discuss the results of experiments on four real world systems evaluated over a time frame of almost four years. Particularly, we have been able to discover in the history of metrics the causes - in the terms of the Granger Test - for 64% to 93% of the defects reported for the systems considered in our experiment [18].

BugMaps: A Tool for the Visual Exploration and Analysis of Bugs To harness the complexity of big legacy software, software engineering tools need more and more information on these systems. This information may come from analysis of the source code, study of execution traces, computing of metrics, etc. One source of information received less attention than source code: the bugs on the system. Little is known about the evolutionary behavior, lifetime, distribution, and stability of bugs. We propose to consider bugs as first class entities and a useful source of information that can answer such topics. Such analysis is inherently complex, because bugs are intangible, invisible, and difficult to be traced. Therefore, our tool extracts information about bugs from bug tracking systems, link this information to other software artifacts, and explore interactive visualizations of bugs that we call bug maps [19].

A Catalog of Patterns for Concept Lattice Interpretation in Software Reengineering Formal Concept Analysis (FCA) provides an important approach in software reengineering for software understanding, design anomalies detection and correction. However, FCA-based approaches have two problems: (i) they produce lattices that must be interpreted by the user according to his/her understanding of the technique and different elements of the graph; and, (ii) the lattice can rapidly become so big that one is overwhelmed by the mass of information and possibilities. We make a catalog of important patterns in concept lattices, which can allow automating the task of lattice interpretation. The approach helps the reengineer to concentrate on the task of reengineering rather than understanding a complex lattice. We provide interpretation of these patterns in a generalized manner and illustrate them on various contexts constructed from program information of different open-source systems. We also present a tool that allows automated extraction of the patterns from concept lattices [16].

5.4. Reflective

Participants: Marcus Denker [Correspondant], Stéphane Ducasse.

DynamicSchema: a lightweight persistency framework for context-oriented data management While context-oriented programming technology so far has focused mostly on behavioral adaptation, context-oriented data management has received much less attention. We make a case for the problem of context-oriented data management, using a concrete example of a mobile application. We illustrate some of the issues involved and propose a lightweight persistency framework, called DynamicSchema, that resolves some of these issues. The solution consists in a flexible reification of the database schema, as a convenient dynamic data structure that can be adapted at execution time, according to sensed context changes. Implementing our mobile application using this framework enabled us to reduce the complexity of the domain modeling layer, to facilitate the production of code with low memory footprint, and to simplify the implementation of certain scenarios related to context-dependent security concerns [17].

6. Bilateral Contracts and Grants with Industry

6.1. Resilience FUI

Participants: Nicolas Petton [Correspondant], Stéphane Ducasse, Damien Cassou.

Contracting parties: Nexedi, Morphom Alcatel-Lucent Bell Labs, Astrium Geo Information, Wallix, XWiki, Alixen, Alterway, Institut Télécom, Université Paris 13, CEA LIST.

Resilience's goal is to protect private data on the cloud, to reduce spying and data loss in case of natural problems. Resilience propose to develop a decentralized cloud architecture: SafeOS. Safe OS is based on replication of servers. In addition a safe solution for documents should be developed. Sandboxing for Javascript applications should be explored.

There is a plethora of research articles describing the deep semantics of JavaScript. Nevertheless, such articles are often difficult to grasp for readers not familiar with formal semantics. In our first report, we propose a digest of the semantics of JavaScript centered around security concerns. This report proposes an overview of the JavaScript language and the misleading semantic points in its design. The first part of the report describes the main characteristics of the language itself. The second part presents how those characteristics can lead to problems. The document finishes by showing some coding patterns to avoid certain traps and presents some ECMAScript 5 new features.

6.2. Generali Belgium

Participants: Nicolas Anquetil [Correspondant], Stéphane Ducasse, Guillaume Larcheveque, Muhammad Bhatti, Camille Teruel.

Contracting parties:

Synectique our startup company project;

Generali Assurances http://www.generali.be.

RMoD is looking into providing a software solution to Generali Belgium for its software maintenance and reengineering problems. The goal is to support decision making in a software migration project. The partner needs tools for parsing their legacy code (in a specific, not well-known language) and help in identifying dead code or code duplication. This should serve as an essential element of decision support in the partner's software migration project.

The contract is worth 30.000€.

6.3. Pharo Consortium

We launched the Pharo Consortium. Over 25 companies are interested in participating. Inria supports the consortium with one full time engineer starting in 2011.

7. Partnerships and Cooperations

7.1. Regional Initiatives

We have signed a convention with the CAR team led by Noury Bouraqadi of École des Mines de Douai. In such context we co-supervized two PhD students (Mariano Martinez-Peck and Nick Papoylias). The team is also an important contributor and supporting organization of the Pharo project.

7.2. National Initiatives

7.2.1. ANR

7.2.1.1. Cutter

Participants: Stéphane Ducasse [Correspondant], Nicolas Anquetil, Damien Pollet, Muhammad Bhatti, Andre Hora.

This parternship is done with the following members from the LIRMM-D'OC-APR: Marianne Huchard, Roland Ducournau, Jean-Claude König, Rodokphe Giroudeau, Abdelhak-Djamel Seriai, and Rémi Watrigant.

CUTTER is a Basic Research project that addresses the problems of object-oriented system (re-)modularization by developing, combining, and evaluating new techniques for analyzing and modularizing code. In particular, it will: (i) use concurrently and collaboratively four package decomposition techniques; and (ii) take into account different levels of abstractions (packages, classes).

7.3. European Initiatives

Participants: Stéphane Ducasse [correspondant], Veronica Uquillas-Gomez, Marcus Denker.

7.3.1. IAP MoVES

Participant: Stéphane Ducasse [correspondant].

The Belgium IAP (Interuniversity Attraction Poles) MoVES (Fundamental Issues in Software Engineering: Modeling, Verification and Evolution of Software) is a project whose partners are the Belgium universities (VUB, KUL, UA, UCB, ULB, FUNDP, ULg, UMH) and three European institutes (Inria, IC and TUD) respectively from France, Great Britain and Netherlands. This consortium combines the leading Belgian research teams and their neighbors in software engineering, with recognized scientific excellence in MDE, software evolution, formal modeling and verification, and AOSD. The project focusses on the development, integration and extension of state-of-the-art languages, formalisms and techniques for modeling and verifying dependable software systems and supporting the evolution of Software-intensive systems. The project has started in January 2007 and is scheduled for a 60-months period. Read more at http://moves.vub.ac.be.

7.3.2. ERCIM Software Evolution

We are involved in the ERCIM Software Evolution working group since its inception. We participated at his creation when we were at the University of Bern.

7.4. International Initiatives

7.4.1. Inria Associate Teams

7.4.1.1. PLOMO

Title: Customizable Tools and Infrastructure for Software Development and Maintenance
Inria principal investigator: Stéphane Ducasse
International Partner (Institution - Laboratory - Researcher):
University of Chile (Chile) - PLEIAD
Duration: 2011–2013
See also: http://pleiad.dcc.uchile.cl/research/plomo

Project Description

Software maintenance is the process of maintaining a software system by removing bugs, fixing performance issues and adapting it to keep it useful and competitive in an ever-changing environment [32]. Performing effective software maintenance and development is best achieved with effective tool support, provided by a variety of tools, each one presenting a specific kind of information supporting the task at hand [34]. The goal of PLOMO is to develop new meta tools to improve and bring synergy in the existing infrastructure of Pharo (for software development) and the Moose software analysis platform (for software maintenance).

PLOMO will (1) enhance the Opal open compiler infrastructure to support plugin definition, (2) offer an infrastructure for change and event tracking as well as model to compose and manipulate them, (3) work on a layered library of algorithms for the Mondrian visualization engine of Moose, (4) work on new ways of profiling applications. All the efforts will be performed on Pharo and Moose, two platforms heavily used by the RMoD and PLEIAD team.

The outcomes of PLOMO will include new research advances in the field of (i) bytecode generation for dynamic language; (ii) change and event tracking; (iii) software visualization engine; (iv) agile profiling framework. These four topics are recurrently considered by the most prestigious and competitive conferences (e.g., ECOOP, OOPSLA, FSE, ESEC, ICSE, TOOLS) and journals (e.g., TSE, TOPLAS, ASE), to which the participants of the PLOMO project are used to publish.

A strong focus on publishing our results in relevant scientific forum will remain a top priority. The artifacts produced by PLOMO will strongly reinforce the Pharo programming language and the Moose software analysis platform. The development and progress of Pharo is structured by RMoD, which has successfully created a strong and dynamic community. Moose is being used to realize consulting activities and it is used as a research platform in about 10 Universities, worldwide. We expect PLOMO to have a strong impact in both the software products and the communities structured around them.

Research Visits to Chile

- Benjamin van Ryseghem from May 28th until June 16th, 2012.
- Damien Pollet from November 1st until November 30th, 2012.
- Marcus Denker from November 5th until November 22nd, 2012

Recent Results

In the second year of execution of Plomo, work has focused on:

- Rizel: a performance evolution monitor.
- A book chapter on Roassal in the book Pharo By Example 2
- Roassal also won the third place award in the ESUG 2012 innovation technology awards.
- Athens, the graphic rendering engine developed by RMoD, is used by Roassal.
- Starting of the founding process of Synectique, a company based in Lille that offers solutions based on the Moose platform. ObjectProfile offers to Synectique a dedicated support of Roassal.
- Integration of profiling techniques into Jenkins, the continuous integration server used for Pharo. We expect to have a massive amount of profiling information.
- Opal debugging and development continued. The bytecode backend is ready for integration in Pharo 2.0.
- Gradualtalk: a gradually typed Smalltalk, built on Opal, has been implemented. It allows code in Pharo to be gradually and optionally typed.
- The Announcements framework to enable change and event tracking.
- Spec: a Framework for the Specification and Reuse of UIs and their Models. It uses the Announcements framework to enable fine-grained UI refreshes. Roassal makes use of Spec for its component
- Work on the DIE domain-specific language and the definition of IDE plugins using it, as well as work on change prediction models are still ongoing.

Supervised PhD students

- Vanessa Peña, PhD student Universidad de Chile. She is working on test coverage and domain specific analyses
- Juan Pablo Sandoval, PhD student Universidad de Chile.

Companies Using our Results

- Synectique is a company delivering dedicated software analysis. Synectique uses Roassal to visually report the analysis of customer source code. The founding process started in 2012, and is expected to be finished in 2013.
- ObjectProfile was founded in 2011 in Chile. Its business plan is essentially focused on Pharo and Roassal. Object Profile offers support of its products to RMoD and Synectique. A number of features of Roassal have been designed to meet Synectique's requirements (e.g., the navigation and scrolling options).

Publications

- Benjamin Van Ryseghem, Stéphane Ducasse, Johan Fabry, Spec: a Framework for the Specification and Reuse of UIs and their Models, in Proceedings of the 4th International Workshop on Smalltalk Technologies (IWST'12), Collocated with ESUG, August 2012. ACM Digital Library (To Appear). [20]
- Juan Pablo Sandoval, Tracking Down Software Changes Responsible for Performance Loss, in Proceedings of the 4th International Workshop on Smalltalk Technologies (IWST'12), Collocated with ESUG, August 2012. ACM Digital Library (To Appear)

7.4.2. Participation In International Programs

7.4.2.1. Project Pequi – Inria/CNPq Brésil

The Pequi project is a collaboration between Professor Marco T. Valente's team at the Federal University of Minas Gerais in Brazil and the RMoD team. It focuses in producing Metrics, Techniques, and Tools for Software Remodularization.

It is recognized that software systems must be continuously maintained and evolved to remain useful. However, ongoing maintenance over the years contributes to degrade the quality of a system. Thus reengineering activities, including remodularization activities, are necessary to restore or enhance the maintainability of the systems. To help in the remodularization of software systems, the project will be structured in two main research lines in which both teams have experience and participation: (i) Evaluation and Characterization of Metrics for Software Remodularization; and (ii) Tools and Techniques for Removal of Architectural Violations.

The project started in July 2011 with a visit of Dr. Nicolas Anquetil to the brazilian team. The project will last 24 months.

Research Visits

- Nicolas Anquetil, from August 6th to 11th.
- Andre Hora, from November 26th to January 4th.

7.4.3. Others

We are building an ecosystem around Pharo with international research groups, universities and companies. Several research groups (such as Software Composition Group – Bern, and Pleaid – Santiago) are using Pharo. Many universities are teaching OOP using Pharo and its books. Several companies worldwide are deploying business solutions using Pharo.

7.5. International Research Visitors

7.5.1. Visits of International Scientists

In the context of the PLOMO associated Team with the University of Chile:

- Johan Fabry from March 19th until March 23rd, 2012
- Johan Fabry from August 17th until Sept 2nd, 2012.
- Juan Pablo Sandoval from 9 November until 2 December 2012. The topic of the research visit is monitoring of performance evolution.

In the context of the Pequi project associated Team with the Federal University of Minas Gerais:

- Professor Marco Tulio Valente visited from February 7th to 13th.
- Ricardo Terra PhD student visited us for one week in begining of April 2012.

Other visits of international scientists:

- Fernando Olivero, PhD Student from the University of Lugano, Switzerland, visited RMoD in March 2012.
- Jurgen VinJu, group leader of SEN1 - Software Analysis & Transformation at CWI, visited us on May 10th and 11th.

7.5.1.1. Internships

Ezequiel La Mónica (from Apr 2012 until Jun 2012)

Subject: Rule checking for pharo

Institution: University of Buenos Aires (Argentina)

Cesar Couto (from December 2011 until February 2012)

Subject: Uncovering Causal Relationships between Software Metrics and Bugs

Institution: Federal University of Minas Gerais, Brazil

7.5.2. Visits to International Teams

In the context of the PLOMO associated Team with the University of Chile:

- Marcus Denker from January 17th to February 1st.
- Benjamin van Ryseghem from May 28th to June 16th.
- Damien Pollet from October 31st to November 13th.
- Marcus Denker from November 5th to November 22nd.

In the context of the Pequi project associated Team with the Federal University of Minas Gerais:

- Nicolas Anquetil, from August 4th to 19th.
- Andre Hora, from November 26th to January 4th.

Many RMoDmembers did various visits at many occasions to, *e.g.*,, Bruxelles in Belgium, Cologne in Germany, Gand in Belgium, Bern in Switzerland, Riva del Garda, Italy, and Belo Horizonte in Brazil.

8. Dissemination

8.1. Scientific Animation

8.1.1. Examination Committees

Stéphane Ducasse was in the examination committee of the following PhD theses:

- *Supporting Integration Activities in Object-Oriented Applications*, Veronica Uquillas-Gomez, Vrije Universiteit Brussel, Belgium. 04/10/12 (advisor)
- *Application-Level Virtual Memory for Object-Oriented Systems*, Mariano Martinez-Peck. Ecole des Mines de Douai. October 29th (advisor).
- *Analyse statique et dynamique de code et visualisation des logiciels via la m ´etaphore de la ville*, Pierre Caserta, Université de Nancy, France. 7/12/12.
- *Efficient Object Versioning for Object-Oriented Languages from Model to Language Integration*, Frédéric Pluquet, Université Libre de Bruxelles, Belgium France. 3/07/12. (referee)

Stéphane Ducasse was in the examination committee of the following HDR:

- *Faciliter la vérification et la validation des méta-modèles : une approche agile, outillée et orientée données*, Dr. Alain Plantec, Université de Bretagne Occidentale, Brest, 28/11/12.
- *MDE 2.0:Pragmatic formal model verification and other challenges*, Jordi Cabot, Université de Rennes, France. 12/10/12.

Nicolas Anquetil was in the examination committee of the following PhD theses:

- *Supporting Integration Activities in Object-Oriented Applications*, Veronica Uquillas-Gomez, Vrije Universiteit Brussel, Belgium. 04/10/12 (advisor)
- *Analyse et conception d'un modèle de qualité logiciel*, Karine Mordal, Université Vincennes – Saint-Denis – Paris 8, France. 03/12/2012 (reviewer)

Marcus Denker was in the examination committee of the following PhD theses:

- *Bridging the Gap between Machine and Language using First-Class Building Blocks*, Toon Verwaest. University of Bern, Switzerland. 12/03/2012 (reviewer).
- *Application-Level Virtual Memory for Object-Oriented Systems*, Mariano Martinez-Peck. Ecole des Mines de Douai. October 29th (reviewer).

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

Licence : Simon Allier, Programmation d'interface graphique, 46 hours, L2, Université Lille 1 (IUT-A), France

Licence : Nicolas Anquetil, Programmation d'interface graphique, 64 hours, L2, Université Lille 1 (IUT-A), France

Licence : Nicolas Anquetil, Projet final, 16 hours, L2, Université Lille 1 (IUT-A), France

Licence : Damien Cassou, conception orientée objet, 42 hours, L3 Miage, Université Lille 1, France

Master : Damien Cassou, conception objet avancée, 42 hours, M1, Université Lille 1, France

Master : Damien Cassou, maintenance de grands logiciels, 8 hours, M2, Université Lille 1, France

Master : Damien Cassou, Metaprogramming and Reflection in Common Lisp, 2 hours, M2, University of Potsdam, Germany

Master : Christophe Demarey, Architectures Logicielles, 30 hours, GIS4, Polytech'Lille, France

Master : Marcus Denker, Reflection and Context, 1.5 hours, M2, Universite catholique de Louvain, Belgique

Master : Stéphane Ducasse, MetaModeling, 16 hours, M2, Ecole des Mines de Douai, France

Licence : Stéphane Ducasse, MetaModeling, 6 hours, L3, Université de Lille, France

Employed Engineers : Stéphane Ducasse, Advanced Object-Oriented Design, 9 hours, engineers, Inria engineer formation, France

Master : Stéphane Ducasse, Advanced Object-Oriented Design, 21 hours, M1, Université de Savoie, France

Licence : Stéphane Ducasse, Object-Oriented Design, 4.5 hours, L2, Université de Lille (IUT-A), France

Licence : Anne Etien, Bases de données et Analyse Informatique, 60 hours, L3, Polytech Lille, France

Licence : Anne Etien, Structures de données, 14 hours, L3, Polytech Lille, France

Master : Anne Etien, Programmation par objets, 35 hours, M1, Polytech Lille, France

Master : Anne Etien, Systèmes d'information objets, 10 hours, M1, Polytech Lille, France

Master : Anne Etien, Bases de données, 30 hours, M1, Polytech Lille, France
Master : Anne Etien, Ingénierie Logicielle, 20 hours, M2, Polytech Lille, France
Licence : Damien Pollet, Challenge création entreprise, 8 hours, M1, Telecom Lille 1, France
Licence : Damien Pollet, Technologies des systèmes d'information ouverts, 27 hours, L3, Telecom Lille 1, France
Licence : Damien Pollet, Introduction à l'algorithmique, 9 hours, L1, Telecom Lille 1, France
Licence : Damien Pollet, Conception et programmation orientée objet, 166 hours, L3, Telecom Lille 1, France
Licence : Damien Pollet, Architecture des ordinateurs, 4.5 hours, L3, Telecom Lille 1, France
Master : Damien Pollet, Ingénierie logicielle, 2 hours, M1, Telecom Lille 1, France
Licence : Igor Stasenko, Vector graphics, 1 hour, L3, Argentina

8.2.2. Supervision

PhD & HdR (Les thèses soutenues doivent figurer dans la bibliographie) :

PhD : Mariano Martinez-Peck, Application-Level Virtual Memory for Object-Oriented Systems, Ecole des Mines de Douai, October 29th 2012, Stéphane Ducasse, Marcus Denker
PhD : Veronica Uquillas-Gomez, Supporting Integration Activities in Object-Oriented Applications, Vrije Universiteit Brussel, October 4th 2012, Stéphane Ducasse, Nicolas Anquetil
PhD in progress : Camillo Bruni, no title yet, 2011-03-01, Stéphane Ducasse, Marcus Denker
PhD in progress : Andre Hora, Improving Static Analysis with Domain-Specific Rules, 01/12/2011, Nicolas Anquetil, Stéphane Ducasse
PhD in progress : Camille Teruel, Security for dynamic languages, 01/12/2012, Stéphane Ducasse, Damien Cassou
PhD in progress : Nick Papoylias, Languages and Development Environments for Mobile Autonomous Robots, October 1st 2010, Marcus Denker, Stéphane Ducasse
PhD in progress : Jean Baptiste Arnaud, Towards First Class References as a Security Infrastructure in Dynamic Languages, September 1st 2009, Marcus Denker, Stéphane Ducasse
PhD in progress : Martin Dias, Supporting Merging, December 3rd 2012, Damien Cassou, Stéphane Ducasse
PhD in progress : Guillermo Polito, Isolation and Reflection in Dynamic Object Oriented Languages, 01-04-2012, Noury Bouraqadi, Luc Fabrese, Marcus Denker, Stéphane Ducasse

8.3. Popularization

- Nicolas Anquetil presented SourceCity (an integrated environment for software analysis, in which software systems are visualized as interactive, navigable 3D cities) at several occasions.
- Nicolas Anquetil presented Moose during the RIC days, which introduce research to master students.
- Muhammad Bhatti discussed about how to create a startup company during the RIC days.
- Muhammad Bhatti presented to PhD students how to promote research results.
- Damien Cassou received a student from *Collège (3ème)* to present research activities and software development.
- Marcus Denker, Esteban Lorenzano and Stéphane Ducasse will present at fOSSa 2012 in December.
- Marcus Denker and Stéphane Ducasse gave presentations about Pharo at the open source conference FOSDEM 2012.

9. Bibliography

Major publications by the team in recent years

[1] N. ANQUETIL, K. M. DE OLIVEIRA, K. D. DE SOUSA, M. G. BATISTA DIAS. *Software maintenance seen as a knowledge management issue*, in "Inf. Softw. Technol.", 2007, vol. 49, n^{o} 5, p. 515–529, http://dx.doi.org/10.1016/j.infsof.2006.07.007.

[2] N. ANQUETIL, T. LETHBRIDGE. *Comparative study of clustering algorithms and abstract representations for software remodularization*, in "IEE Proceedings - Software", 2003, vol. 150, n° 3, p. 185-201.

[3] M. DENKER, S. DUCASSE, É. TANTER. *Runtime Bytecode Transformation for Smalltalk*, in "Journal of Computer Languages, Systems and Structures", July 2006, vol. 32, n° 2-3, p. 125–139 [*DOI :* 10.1016/J.CL.2005.10.002], http://scg.unibe.ch/archive/papers/Denk06aRuntimeByteCodeESUGJournal.pdf.

[4] S. DUCASSE, T. GÎRBA, A. KUHN, L. RENGGLI. *Meta-Environment and Executable Meta-Language using Smalltalk: an Experience Report*, in "Journal of Software and Systems Modeling (SOSYM)", February 2009, vol. 8, n° 1, p. 5–19 [*DOI :* 10.1007/S10270-008-0081-4], http://scg.unibe.ch/archive/drafts/Duca08a-Sosym-ExecutableMetaLanguage.pdf.

[5] S. DUCASSE, M. LANZA. *The Class Blueprint: Visually Supporting the Understanding of Classes*, in "Transactions on Software Engineering (TSE)", January 2005, vol. 31, n° 1, p. 75–90 [*DOI :* 10.1109/TSE.2005.14], http://scg.unibe.ch/archive/papers/Duca05bTSEClassBlueprint.pdf.

[6] S. DUCASSE, A. LIENHARD, L. RENGGLI. *Seaside: A Flexible Environment for Building Dynamic Web Application*, in "IEEE Software", 2007, vol. 24, n° 5, p. 56–63, http://dx.doi.org/10.1109/MS.2007.144.

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

[8] S. DUCASSE, D. POLLET. *Software Architecture Reconstruction: A Process-Oriented Taxonomy*, in "IEEE Transactions on Software Engineering", July 2009, vol. 35, n° 4, p. 573-591 [*DOI :* 10.1109/TSE.2009.19], http://scg.unibe.ch/archive/external/Duca09x-SOAArchitectureExtraction.pdf.

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

[10] N. SCHÄRLI, A. P. BLACK, S. DUCASSE. *Object-oriented Encapsulation for Dynamically Typed Languages*, in "Proceedings of 18th International Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'04)", October 2004, p. 130–149 [*DOI :* 10.1145/1028976.1028988], http://scg.unibe.ch/archive/papers/Scha04bOOEncapsulation.pdf.

Publications of the year

Articles in International Peer-Reviewed Journal

[11] M. DIAS, M. MARTINEZ PECK, S. DUCASSE, G. ARÉVALO. *Fuel: A Fast General Purpose Object Graph Serializer*, in "Software: Practice and Experience", June 2012, http://hal.inria.fr/hal-00703574.

[12] J. LAVAL, N. ANQUETIL, M. U. BHATTI, S. DUCASSE. *OZONE: Layer Identification in the presence of Cyclic Dependencies*, in "Science of Computer Programming", September 2012, http://hal.inria.fr/hal-00732655.

[13] J. LAVAL, S. DUCASSE. *Resolving cyclic dependencies between packages with Enriched Dependency Structural Matrix*, in "Software - Practice and Experience", November 2012, http://hal.inria.fr/hal-00748120.

[14] J. LAVAL, J.-R. FALLERI, P. VISMARA, S. DUCASSE. *Efficient Retrieval and Ranking of Undesired Package Cycles in Large Software Systems*, in "Journal of Object Technology", April 2012, vol. 11, n° 1 [*DOI :* 10.5381/JOT.2012.11.1.A4], http://hal.inria.fr/hal-00692569.

International Conferences with Proceedings

[15] S. ALLIER, A. HORA, N. ANQUETIL, S. DUCASSE. *A Framework to Compare Alert Ranking Algorithms*, in "19th Working Conference on Reverse Engineering", Kingston, Canada, October 2012, p. 277-285, http://hal.inria.fr/hal-00747817.

[16] M. U. BHATTI, N. ANQUETIL, M. HUCHARD, S. DUCASSE. *A Catalog of Patterns for Concept Lattice Interpretation in Software Reengineering*, in "SEKE 2012: 24th International Conference on Software Engineering & Knowledge Engineering", San Francisco Bay, United States, D. ZHANG, M. REFORMAT, S. GOKHALE, J. C. MALDONADO (editors), Knowledge Systems Institute Graduate School, July 2012, p. 118-124, http://hal.inria.fr/hal-00700046.

[17] S. CASTRO, S. GONZÁLEZ, K. MENS, M. DENKER. *DynamicSchema: a lightweight persistency framework for context-oriented data management*, in "COP '12", Beijing, China, M. APPELTAUER (editor), ACM, June 2012, p. 5:1–5:6 [*DOI :* 10.1145/2307436.2307441], http://hal.inria.fr/hal-00720348.

[18] C. COUTO, S. CHRISTOFER, M. TULIO VALENTE, R. BIGONHA, N. ANQUETIL. *Uncovering Causal Relationships between Software Metrics and Bugs*, in "CSMR - European Conference on Software Maintenance and Reengineering - 2012", Szeged, Hungary, IEEE Comp. Soc., 2012, http://hal.inria.fr/hal-00668151.

[19] A. HORA, N. ANQUETIL, S. DUCASSE, M. U. BHATTI, C. COUTO, M. TULIO VALENTE, J. MARTINS. *BugMaps: A Tool for the Visual Exploration and Analysis of Bugs*, in "Proceedings of the 16th European Conference on Software Maintenance and Reengineering (CSMR'12) - Tool Demonstration Track", Szeged, Hungary, 2012, page : to appear, http://hal.inria.fr/hal-00668397.

[20] B. VAN RYSEGHEM, S. DUCASSE, J. FABRY. *Spec: A Framework for the Specification and Reuse of UIs and their Models*, in "Proceedings of ESUG International Workshop on Smalltalk Technologies (IWST 2012)", Gent, Belgique, August 2012, http://hal.inria.fr/hal-00759030.

Conferences without Proceedings

[21] A. AUTHOSSERRE-CAVARERO, F. BERTRAND, M. BLAY- FORNARINO, P. COLLET, H. DUBOIS, S. DUCASSE, S. DUPUY-CHESSA, C. FARON-ZUCKER, C. FAUCHER, J.-Y. LAFAYE, P. LAHIRE, O. LE GOAER, J. MONTAGNAT, A.-M. PINNA-DERY. *Interopérabilité des systèmes d'information : approches dirigées par les modèles*, in "Inforsid", Montpellier, France, May 2012, http://hal.inria.fr/hal-00707536.

Scientific Books (or Scientific Book chapters)

[22] H. VERJUS, S. CIMPAN, I. ALLOUI. *An Architecture-Centric Approach for Information System Architecture Modeling, Enactement and Evolution*, in "Innovative Information Systems Modelling Techniques", InTech, May 2012, p. 15-46, http://hal.inria.fr/hal-00702521.

References in notes

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

[24] A. BERGEL, S. DUCASSE, O. NIERSTRASZ. *Classbox/J: Controlling the Scope of Change in Java*, in "Proceedings of 20th International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'05)", New York, NY, USA, ACM Press, 2005, p. 177–189 [*DOI :* 10.1145/1094811.1094826], http://scg.unibe.ch/archive/papers/Berg05bclassboxjOOPSLA.pdf.

[25] A. BERGEL, S. DUCASSE, O. NIERSTRASZ, R. WUYTS. *Stateful Traits*, in "Advances in Smalltalk — Proceedings of 14th International Smalltalk Conference (ISC 2006)", LNCS, Springer, August 2007, vol. 4406, p. 66–90, http://dx.doi.org/10.1007/978-3-540-71836-9_3.

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

[27] A. P. BLACK, N. SCHÄRLI, S. DUCASSE. *Applying Traits to the Smalltalk Collection Hierarchy*, in "Proceedings of 17th International Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'03)", October 2003, vol. 38, p. 47–64 [*DOI :* 10.1145/949305.949311], http://scg.unibe.ch/archive/papers/Blac03aTraitsHierarchy.pdf.

[28] E. BODDEN, A. SEWE, J. SINSCHEK, H. OUESLATI, M. MEZINI. *Taming Reflection*, in "ICSE", 2011.

[29] G. BRACHA, D. UNGAR. *Mirrors: design principles for meta-level facilities of object-oriented programming languages*, in "Proceedings of the International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'04), ACM SIGPLAN Notices", New York, NY, USA, ACM Press, 2004, p. 331–344, http://bracha.org/mirrors.pdf.

[30] D. CAROMEL, J. VAYSSIÈRE. *Reflections on MOPs, Components, and Java Security*, in "ECOOP '01: Proceedings of the 15th European Conference on Object-Oriented Programming", Springer-Verlag, 2001, p. 256–274.

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

[32] E. CHIKOFSKY, J. CROSS II. *Reverse Engineering and Design Recovery: A Taxonomy*, in "IEEE Software", January 1990, vol. 7, n° 1, p. 13–17, http://dx.doi.org/10.1109/52.43044.

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

[34] S. DEMEYER, S. DUCASSE, O. NIERSTRASZ. *Object-Oriented Reengineering Patterns*, Morgan Kaufmann, 2002, http://www.iam.unibe.ch/~scg/OORP.

[35] S. DENIER. *Traits Programming with AspectJ*, in "Actes de la Première Journée Francophone sur le Développement du Logiciel par Aspects (JFDLPA'04)", Paris, France, P. COINTE (editor), September 2004, p. 62–78.

[36] S. DUCASSE, T. GÎRBA. *Using Smalltalk as a Reflective Executable Meta-Language*, in "International Conference on Model Driven Engineering Languages and Systems (Models/UML 2006)", Berlin, Germany, LNCS, Springer-Verlag, 2006, vol. 4199, p. 604–618 [*DOI :* 10.1007/11880240_42], http://scg.unibe.ch/archive/papers/Duca06dMOOSEMODELS2006.pdf.

[37] S. DUCASSE, T. GÎRBA, M. LANZA, S. DEMEYER. *Moose: a Collaborative and Extensible Reengineering Environment*, in "Tools for Software Maintenance and Reengineering", Milano, RCOST / Software Technology Series, Franco Angeli, Milano, 2005, p. 55–71, http://scg.unibe.ch/archive/papers/Duca05aMooseBookChapter.pdf.

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

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

[40] A. DUNSMORE, M. ROPER, M. WOOD. *Object-Oriented Inspection in the Face of Delocalisation*, in "Proceedings of ICSE '00 (22nd International Conference on Software Engineering)", ACM Press, 2000, p. 467–476.

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

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

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

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

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

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

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

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

[49] R. KOSCHKE. *Atomic Architectural Component Recovery for Program Understanding and Evolution*, Universität Stuttgart, 2000, http://www.informatik.uni-stuttgart.de/ifi/ps/bauhaus/papers/koschke.thesis.2000.html.

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

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

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

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

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

[55] M. S. MILLER, C. MORNINGSTAR, B. FRANTZ. *Capability-based Financial Instruments*, in "FC '00: Proceedings of the 4th International Conference on Financial Cryptography", Springer-Verlag, 2001, vol. 1962, p. 349–378.

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

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

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

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

[60] J. H. SALTZER, M. D. SCHOROEDER. *The Protection of Information in Computer Systems*, in "Fourth ACM Symposium on Operating System Principles", IEEE, September 1975, vol. 63, p. 1278–1308.

[61] N. SANGAL, E. JORDAN, V. SINHA, D. JACKSON. *Using Dependency Models to Manage Complex Software Architecture*, in "Proceedings of OOPSLA'05", 2005, p. 167–176.

[62] N. SCHÄRLI, A. P. BLACK, S. DUCASSE. *Object-oriented Encapsulation for Dynamically Typed Languages*, in "Proceedings of 18th International Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'04)", October 2004, p. 130–149 [*DOI :* 10.1145/1028976.1028988], http://scg.unibe.ch/archive/papers/Scha04bOOEncapsulation.pdf.

[63] N. SCHÄRLI, S. DUCASSE, O. NIERSTRASZ, A. P. BLACK. *Traits: Composable Units of Behavior*, in "Proceedings of European Conference on Object-Oriented Programming (ECOOP'03)", LNCS, Springer Verlag, July 2003, vol. 2743, p. 248–274 [*DOI :* 10.1007/B11832], http://scg.unibe.ch/archive/papers/Scha03aTraits.pdf.

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

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

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

[67] D. VAINSENCHER. *MudPie: layers in the ball of mud.*, in "Computer Languages, Systems & Structures", 2004, vol. 30, n^{o} 1-2, p. 5–19.

[68] N. WILDE, R. HUITT. *Maintenance Support for Object-Oriented Programs*, in "IEEE Transactions on Software Engineering", December 1992, vol. SE-18, n^{o} 12, p. 1038–1044.