PLOMO Associate Team Final Report

A. Bergel S. Ducasse M. Denker J. Fabry

23 September 2013

Associate Team acronym: PLOMO

Period of activity: 2010-2013

Principal investigator (Inria): S. Ducasse – RMoD – Lille Nord Europe

Principal investigator (partner): A. Bergel – DCC - Pleiad – University of Chile.

Other participants: M. Denker (RMoD), J. Fabry (DCC), D. Pollet (RMoD), R. Robbes (DCC).

1 Goals

The goal of Plomo is to develop new meta tools to improve and bring synergy in the existing infrastructure of Pharo[1] (for software development) and the Moose software analysis platform[2] (for maintenance). PLOMO will

- enhance the Opal open compiler infrastructure to support plugin definition,
- offer an infrastructure for change and event tracking as well as models to compose and manipulate them,
- work on a layered library of algorithms for the Mondrian visualization engine of Moose,
- 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 teams. Below is a description of the four work packages as initially formulated.

Work Package One: Opal Open Compiler Infrastructure. Opal is a new, extensible and modular compiler framework. The front-end is based on the notion of dynamically composable parsers [21] and a high-level code representation based on an AST (abstract syntax tree) with code transformation support [23]. The backend is based on an intermediate representation of byte code. Such a high-level representation of byte code offers a good possibility to express advanced, on the fly, modification of methods at the byte code level [7]. A flexible and extensible byte code level infrastructure is needed to introduce code for hooks and low level probes, as needed by advanced aspect oriented languages (AOP) [16] or dynamic analysis tools like runtime profilers. The compiler framework was developed as a joint project between RMoD and the SCG

[1] http://pharo.org
[2] http://www.moosetechnology.org

at University of Bern/Switzerland. Opal acts as the infrastructure behind several other work packages and needs to be enhanced to enable full support for those.

The goal of this task is to be able to replace the existing compiler infrastructure in place in Pharo by Opal, allowing the other tasks of this proposal to use Opal as a basis. Replacing the old Pharo compiler validates the compiler as practically usable.

Work Package Two: A Domain-Specific Language for Change and Event Tracking. Many development environment tools provide additional support to the fundamental edit-compile-run cycle of development and maintenance. These range from basic, state-of-the-practice tools such syntax highlighting, code autocompletion, or notification of changes, to experimental tools such as task context management [14] and code clone detection and clone management. These tools need to be integrated in the IDE in a fluid manner.

These additional tools hook into the core of the IDE, adding their notification system for the actions taking place inside of the IDE. The events that they track can differ, ranging from events produced by the IDE (changes to the code, navigation in the code, tool usage, copy/paste, etc.) to events produced by the program that is being developed at runtime, as generated by debugging or profiling tools. Currently, these additional tools contain an ad-hoc event handler, i.e., each tool implements its own event handlers and event generators. As a result, there is low reuse in that part of the construction of such additional support tools, and furthermore the different event handlers in the system may interfere in each others' operation. The various event generators have heterogeneous interfaces as well, so that a tool needing to handle several kinds of events is exposed to additional complexity.

The goal of this task is to develop a general event notification system for Pharo and building a set of suitable abstractions on top of this system. We plan to extend and refine the current Announcement system, and reify events at the level of abstraction that allows the additional support tools to express sequences of events of interest in a concise way. Conceptually this is a case of event-based Aspect Oriented Programming [10]. The IDE generates a stream of events which are join points, the abstraction level is obtained by a (higher-level) join point model, and the tools express their interest in a certain kind of stream of events by declaring a pointcut. Developing a Domain-Specific Aspect Language for IDE events will allow tool builders to declare event handlers in a concise and homogeneous way, allowing tools to express interest in different stream of events seamlessly.

Work Package Three: Mondrian 2. Software visualization is an essential component of many reverse engineering and software analysis approaches. The first step for this research axis is to produce the necessary infrastructure for conducting analysis and experiments using software visualization. We envision a framework that is open and malleable for a broad range of visual representations. This work package is built on top of the Mondrian visualization framework. Mondrian has been successfully employed in numerous academic and industrial case studies. It has demonstrated its ability to produce useful and practical high level source code representations, helpful to identify bugs and anomalies in software programs [17, 2, 20]. Although expressive, Mondrian suffers from its built-in graph model to represent data. It is often a constraint since its restrict creativity and scalability (especially when low and high level of details are simultaneously displayed). As the result of this first work package,we plan to produce Mondrian2 (working title for the new version of Mondrian), an open-source framework for agile, flexible, scalable and open visualizations of both static as dynamic information. For the latter, Mondrian will be able to take advantage of the new event mechanism implemented in the second workpackage to allow for dynamic visualizations of behavioral/run-time information.

Work Package Four: New Profiling Approaches. Application profiling can be realized for a variety of reasons (locating performance bottlenecks, assessing test coverage, to support feature location, etc.). Each use case requires different strategies to gather the necessary information, and has different performance requirements.

In this work package we will focus on providing the necessary infrastructure for profiling applications in such a variety of situations. There are several scientific challenges such as collecting, storing events in memory efficient manner.

Such a framework ought to be flexible and designed for rapid prototyping of profilers. The key asset of this framework is to be open for dynamic extension and refinement: profilers will be adjustable while acting on an executing software program. This ability, which we call agility, is essential to enable an effective profiling in a dynamic programming language and environment such as Pharo, the programming language we have chosen for conducting this work. To effectively exploit produced profilers, the framework has to be fully integrated with the IDE of Pharo. As an outcome, this work package will produce an agile and effective framework for designing and realizing program execution profilers.

2 Overview of the activities

The activities of the PLOMO members were mainly oriented towards: (1) scientific publications, (2) software development and (3) book writing.

2.1 Organization

- Workshops at ESUG in 2012 and 2013.
- Coding sprints in 2013

2.2 PhD Students

- Juraj Kubelka, PhD student Universidad de Chile.
- Juan Pablo Sandoval, PhD student Universidad de Chile.
- Martin Dias, PhD student, RMoD, Inria.

RMoD payed M Dehouck as intern to help optimizing Roassal. He was co-supervised by U. Bhatti (RMOD-Synectique) and A. Bergel. M . Dehouck continued his work in the context of the Google Summer of Code and published an article in an international workshop.

2.3 Research Visits

PLOMO enables the participants RMoD (INRIA Lille Nord Europe) and Pleiad (DCC University of Chile) to collaborate by funding their trips.

2011

From RMoD to PLEIAD

- Marcus Denker from November 7th until November 28th, 2011
- Marcus Denker from January 18 until February 2nd, 2012 (2011 budget)

From PLEIAD to RMoD

- Vanessa Peña and Alexandre Bergel, Aug 15 until Aug 20, 2011. From Aug 20 until Aug 28 they attended ESUG, a conference co-organized by RMoD. The topic of this research visit is test coverage and software visualization. Result of the research activity on software visualization and code profiling have been presented at ESUG.
- Romain Robbes from July 18 until July 24, 2011. Partially funded by Pleiad. Esteban Allende from July 19 until October 2, 2011. Esteban's stay was partially funded by the French Embassy in Chile.

2012

From RMoD to PLEIAD

- Benjamin van Ryseghem from May 29th until June 15th, 2012. Partially funded by Pleiad.
- Damien Pollet from November 1 until November 30, 2012. Partially funded by Pleiad.
- Marcus Denker from November 5 until November 22, 2012

From PLEIAD to RMoD

- Johan Fabry from March 19th until March 23rd, 2012.
- Johan Fabry from August 17th until Sept 2nd, 2012. Fully funded by Pleiad
- Juan Pablo Sandoval from 9 November until 2 December 2012. The topic of the research visit is monitoring of performance evolution.
- Alexandre Bergel, Vanessa Peña, Juan Pablo Sandoval, Pablo Estefo from August 24 until September 2.

All participated at ESUG, a conference co-organized by RMoD, where in parallel work has been performed together with RMoD on profiling applications. These trips were fully funded by Pleiad.

2013

From RMoD to PLEIAD

- Stéphane Ducasse from November 4 until November 15, 2013. Partially funded by Pleiad.

From PLEIAD to RMoD

- Johan Fabry on 15th of July, 18th and 19th of September 2013
- Alexandre Bergel from December 12 until December 29, 2013 (to confirm)
- Alejandro Infante from September 13 until September 21, 2013
- Ronie Salgado in January 2014. (to confirm)

3 Scientific achievements

3.1 Work Package One Achievements: Opal Open Compiler Infrastructure

The compiler framework was used to build a type system for Pharo: Gradualtalk [3]. A gradually-typed Smalltalk allows one to enable incremental typing of existing programs. The main design goal of the type system is to support the features of the Smalltalk language, like metaclasses and blocks, live programming, and to accommodate the programming idioms used in practice. We studied a number of existing projects to determine the features to include in the type system. As a result, Gradualtalk is a practical approach to gradual types in Smalltalk, with a novel blend of type system features that accommodate most programming idioms.

In the context of the Gradualtalk type system, we have further developed, debugged and released the compiler framework. Opal has been integrated as the new compiler for Pharo[5]. It is stable, robust and is the new default compiler for day to day development in Pharo30. Opal provides the basis for many new features in Pharo30 and provides a new foundation for building new layers such as an advanced reflective model.

Opal solves three main problems of the old compiler infrastructure:

- the architecture is not reusable
- the compiler can not be parametrized
- the mapping between source code and bytecode is overly complex.

In essence the work on the compiler framework does not move forward the state of the art in compiler technology however it has already been proven to be a crucial building block. It provides the fundamentals for many research experiments and new features for Pharo. The work on the type system is one example, others are the many new features of Pharo30 that use parts of the new compiler chain, like AST-based navigation in the editor or breakpoints.

The Opal flexible compiler infrastructure will allow us to build a new generation of reflective systems. Such reflective systems are keys to support tools (profilers) and new language design (AOP, proxies, isolation). We are now in a position to perform a new iteration on the reflective layer and metaobject protocol [15] for Pharo. In 2005 we started to develop new reflective foundations and we learned a lot (partial reflection[24], metalevel recursion handling[8], object-centric on the fly propagation[4], proxies [18], isolation [28]). We are now in a situation to step back and redesign the reflective layer based on all the cases we identified and developed.

3.2 Work Package Two Achievements: A Domain-Specific Language for Change and Event Tracking

We developed EPICEA a new model of changes and an implementation representing all the changes made during development [9] - By changes we mean: method, class, package definition, modification, removal but also new coding session, refactorings

It is the basis for a large number of analyses (cherry picking, code review support, replay of sequences, code recovering, browsing in the past) and tools that we will build around change management. In particular EPICEA will be extended to support branch merging and propose new analyses to help developer taking merging decisions.

Complimentary to that, we developed DIE, a Domain-Specific Aspect Language that provides a set of domain-specific abstractions for building plug-ins to a development environment. It allows tool builders to declare event handlers in a concise and homogeneous way, allowing IDE extensions and tools to express interest in different streams of events seamlessly, with a consistent syntax regardless of the source of the event.

3.3 Work Package Three Achievements: Mondrian 2

Over the last 3 years we have developed Roassal, an agile visualization engine. We have dropped the name Mondrian 2 in favor of Roassal. Roassal is made to visualize and interact with arbitrary data, defined in terms of objects and their relationships. Roassal is commonly employed to produce interactive visualizations. The range of applications using Roassal is diverse. For example, the Moose community uses Roassal to visualize software.

Roassal is a visualization engine working on 4 different platforms (Pharo, VisualWorks, Amber, VASmalltalk). Roassal has been adopted by several companies (such as LAM Research[3]) located in Europe, Australia and North America. Maintenance of Roassal and porting efforts are supported by Object Profile[4]. Roassal is a key element of the Moose open-source analysis platform developed by RMOD and the company Synectique[5], a spinoff of RMOD.

Roassal is an engine used by several research group worldwide (including IRC (FR), SCG (CH), Reveal (CH)) and the foundations of several research publications (profiler, memory manager, program visualizations, ...).

The new book *Deep Into Pharo* [6] contains two chapters about Roassal.

3.4 Work Package Four Achievements: New Profiling Approaches

Profiling techniques have been co-developed by RMoD and Pleiad. Understanding the root of a performance drop or improvement requires analyzing different program executions at a fine grain level. Such an analysis involves dedicated profiling and representation techniques. JProfiler and YourKit, two recognized code profilers fail, on both providing adequate metrics and visual representations, conveying a false sense of root course for a performance variation.

We propose performance evolution blueprint, a visual support to precisely compare multiple software executions. Our blueprint is offered by Rizel, a code profiler to efficiently explore performance of a set of benchmarks against multiple software revisions.

3.5 Other Achievements

We worked on Spec: a UI Builder that proposes a new way to compose applications out of widgets [25, 26]. A missing aspect of existing UI Builders is the ability to reuse and compose widget logic. This leads to a significant amount of duplication in UI code. Instead, Spec focuses on widget reuse with widget properties that are defined declaratively and and attached to specific classes that are built specifically to be reusable.

4 Production

4.1 Company Relationships

PLOMO startups

- **Synectique** (`http://www.synectique.eu`) is a company delivering dedicated software analyses. Synectique uses Roassal to visually report customer source code analysis. The founding process started in 2012, and the company was created as a spinoff from Inria in June 2013.

[3] `http://www.lamrc.com`
[4] `http://objectprofile.com`
[5] `http://www.synectique.eu`

- **ObjectProfile** (http://objectprofile.com) 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).

4.2 Software

In addition to OPAL and Roassal, PLOMO members directly worked on Moose and Pharo releases.

Moose (http://www.moosetechnology.org)

We participated to the new release of Moose . The key highlights are:

- Based on Pharo 2.0.
- All built-in visualizations use Roassal.
- Roassal received a significant performance boost and new smart graph layouts like TreeMapLayout or ForceBasedLayout.
- Roassal uses Athens for nice looking vectorial graphics.
- New charting engine based on Roassal: Graph-ET.
- Glamour changed to enable dynamic scripts, while still remaining backward compatible.
- Glamour received a RubricTextPresentation for using the new Rubric text morph.
- Glamour, Roassal, EyeSee, Graph-ET editors are more robust when dealing with errors in scripts.-
- New GTDebugger with dedicated workflows for Announcements, PetitParser and Glamour itself.
- Extended GTInspector for several object types.
- Usable version of GTMetaceller for handling Metacello configurations.
- FAMIX was strengthen to handle functions better in various programming languages.
- New free type fonts and simpler whitespace-loving theme.
- Lower memory footprint for large models.
- Faster MSE import.

Pharo (http://pharo.org)

We participated to the releases of Pharo 1.3, 1.4 and 2.0 and are actively developing Pharo3. The speed of Pharo development is increasing with each version. For Pharo 2.0, a list of changes can be found on the Pharo website[6]. PLOMO members contributed many smaller and larger improvements found while doing the research work described in this report. When visiting,

[6] http://code.google.com/p/pharo/wiki/ActionsInPharo20

PLOMO members take care to organize Pharo Sprints, open meetings focussed on fixing bugs and integrating features.

A small example for how PLOMO directly contributed can be seen with the type system work. While type-checking the code of the Pharo system, a number of problems where detected. Examples where dead code, wrong inheritance relationships between classes and others. All these problems have been fixed in Pharo2.

For Pharo3, PLOMO contributed in a major way: the Opal Compiler replaced the old default compiler infrastructure.

4.3 PLOMO Associated publications

2011

- [11] Stéphane Ducasse, Manuel Oriol, Alexandre Bergel, Challenges to support automated random testing for dynamically typed languages, In Proceedings of the 3rd International Workshop on Smalltalk Technologies (IWST'11), Collocated with ESUG, June 2011. ACM Digital Library http://dx.doi.org/10.1145/2166929.2166938

- [12] Johan Fabry, Andy Kellens, and Stéphane Ducasse. AspectMaps: A Scalable Visualization of Join Point Shadows. In Proceedings of 19th IEEE International Conference on Program Comprehension (ICPC2011), pages 121130. IEEE Computer Society Press, Jul 2011 http://dx.doi.org/10.1109/ICPC.2011.11

2012

- [13] Johan Fabry, Andy Kellens, Simon Denier and Stéphane Ducasse. AspectMaps: Extending Moose to Visualize AOP Software, In Elsevier Science of Computer Programming, Special issue on Experimental Software Toolkits, 2012. http://dx.doi.org/10.1016/j.scico.2012.02.007 (in press).

- [26] 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 http://dx.doi.org/10.1145/2448963.2448965

- [1] 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 http://dx.doi.org/10.1145/2448963.2448966

2013

- [3] Esteban Allende, Oscar Callau, Johan Fabry, Eric Tanter, and Marcus Denker, Gradual Typing for Smalltalk In Science of Computer Programming, accepted and available online. http://dx.doi.org/10.1016/j.scico.2013.06.006

- [27] Juan Pablo Sandoval Alcocer, Alexandre Bergel, Stéphane Ducasse, Marcus Denker Performance Evolution Blueprint: Understanding the Impact of Software Evolution on Performance. Accepted at 1st IEEE Working Conference on Software Visualization VISSOFT, 2013.

- [6] Deep into Pharo, 421 pages, Published by Square Brackets Association http://deepintopharo.com

- [19] Dehouck Mathieu, Usman Bhatti, Alexandre Bergel and Stéphane Ducasse, Pragmatic Visualizations for Roassal: a Florilegium, in International Workshop on Smalltalk Technologies, 2013.
- [9] Martin Dias, Damien Cassou and Stéphane Ducasse, Representing Code History with Development Environment Events, in International Workshop on Smalltalk Technologies, 2013.
- [5] Clément Béra and Marcus Denker, Towards a flexible Pharo Compiler, in International Workshop on Smalltalk Technologies, 2013.

In Preparation and Under Review

- Romain Robbes, Johan Fabry and Marcus Denker, DIE: A Domain Specific Aspect Language for IDE Events, accepted with major revision at Journal of Universal Computer Science.
- Benjamin Van Ryseghem, Stéphane Ducasse, Johan Fabry and Alain Plantec: Seamless reuse of customizable user interfaces with Spec, accepted with minor revisions in Elsevier Science of Computer Programs.
- Romain Robbes, Damien Pollet and Michele Lanza: Improving change prediction by replaying fine-grained development histories, In preparation.

By member of PLOMO

We list publications that have been done by Members of the PLOMO project on related topics outside of the PLOMO collaboration.

- O. Calláu, R. Robbes, É. Tanter, D. Röthlisberger: How (and Why) Developers Use the Dynamic Features of Programming Languages: The Case of Smalltalk, EMSE, in press, Empirical Software Engineering (Springer)
- [22] R. Robbes, D. Röthlisberger, É. Tanter: Extensions during Software Evolution: Do Objects Meet Their Promise, ECOOP 2012, 26th European Conference on Object-Oriented Programming

4.4 PLOMO achievements per year

Now we list the results per year and with different angles.

2011

- Roassal is an agile visualization engine. Roassal is used by the RMoD research group and is is primarily developed in Pharo. Roassal is also used by the Moose software analysis platform.
- Enhanced Opal to support type annotations. Opal is developed and maintained by RMoD.
- ObjectProfile (`http://objectprofile.com`) is a new company based in Chile. Its business plan is essentially focused on Pharo and Roassal.
- Released Pharo 1.3 (`http://www.pharo.org`)

2012

- Rizel: a performance evolution monitor.
- A book chapter on Roassal in the book Deep into Pharo [6]
- Roassal also won the third place award in the ESUG 2012 innovation technology awards.
- Development of Athens the graphic rendering engine developed by RMoD for Pharo. Athens 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.
- Released Pharo 1.4 (http://www.pharo.org)

2013

- GradualTalk Paper accepted at Science of Computer Programming [3].
- Juan Pablo Sandoval Alcocer, Alexandre Bergel, Stéphane Ducasse, Marcus Denker Performance Evolution Blueprint: Understanding the Impact of Software Evolution on Performance. Proceedings of 1st IEEE Working Conference on Software Visualization VISSOFT, 2013. [27]
- The book Deep into Pharo, after 4 years of hard work (http://deepintopharo.com) [6]
- Work on the DIE domain-specific language and the definition of IDE plugins using it was submitted to a journal and is in a second round of revisions.
- Organization of a coding sprint at Santiago in January 2013 (12 participants)
- Participated to Moose 4.8 release (http://www.moosetechnology.org)
- Released Pharo 2.0 (http://www.pharo.org)
- Integrated the Opal Compiler in the Pharo3 development branch.

5 Future of the partnership

We really hope that the team will be prolongated for a second three year period.

The synergy between the two teams is working really well - in terms of exchanges, results and future collaborations.

References

[1] Juan Pablo Sandoval Alcocer. Tracking down software changes responsible for performance loss. In *Proceedings of the International Workshop on Smalltalk Technologies*, IWST '12, pages 3:1–3:7, New York, NY, USA, 2012. ACM.

[2] Julio Ariel Hurtado Alegría, Alejandro Lagos, Alexandre Bergel, and María Cecilia Bastarrica. Software process model blueprints. In *Proceedings of the International Conferences on Software Processes (ICSP'10)*. LNCS Springer Verlag, July 2010. to appear.

[3] Esteban Allende, Oscar Callau, Johan Fabry, Eric Tanter, and Marcus Denker. Gradual typing for smalltalk. *Science of Computer Programming*, 2013.

[4] Jean-Baptiste Arnaud, Marcus Denker, Stéphane Ducasse, Damien Pollet, Alexandre Bergel, and Mathieu Suen. Read-only execution for dynamic languages. In *Proceedings of the 48th International Conference Objects, Models, Components, Patterns (TOOLS'10)*, Malaga, Spain, June 2010.

[5] Clément Béra and Marcus Denker. Towards a flexible pharo compiler. In *International Workshop on Smalltalk Technologies 2013*, 2013.

[6] Alexandre Bergel, Damien Cassou, Stéphane Ducasse, and Jannik Laval. *Deep Into Pharo*. Square Bracket Associates, 2013.

[7] Marcus Denker, Stéphane Ducasse, and Éric Tanter. Runtime bytecode transformation for Smalltalk. *Journal of Computer Languages, Systems and Structures*, 32(2-3):125–139, July 2006.

[8] Marcus Denker, Mathieu Suen, and Stéphane Ducasse. The meta in meta-object architectures. In *Proceedings of TOOLS EUROPE 2008*, volume 11 of *LNBIP*, pages 218–237. Springer-Verlag, 2008.

[9] Martín Dias, Damien Cassou, and Stéphane Ducasse. Representing code history with development environment events. In *International Workshop on Smalltalk Technologies 2013*, 2013.

[10] Rémi Douence, Pascal Fradet, and Mario Südholt. A framework for the detection and resolution of aspect interactions. In *GPCE'02: Proceedings of the 1st International Conference on Generative Programming and Component Engineering*, pages 173–188. Springer-Verlag, 2002.

[11] Stéphane Ducasse, Manuel Oriol, and Alexandre Bergel. Challenges to support automated random testing for dynamically typed languages. In *Proceedings of ESUG International Workshop on Smalltalk Technologies (IWST 2011)*, Edinburgh, Scotland, 2011.

[12] Johan Fabry, Andy Kellens, Simon Denier, and Stéphane Ducasse. AspectMaps: A scalable visualization of join point shadows. In *Proceedings of the 19th International Conference on Program Comprehension*, ICPC'11, pages 121–130. IEEE Computer Society Press, 2011.

[13] Johan Fabry, Andy Kellens, Simon Denier, and Stéphane Ducasse. Aspectmaps: Extending moose to visualize aop software. *Science of Computer Programming*, 2012. to appear.

[14] Mik Kersten and Gail C. Murphy. Using task context to improve programmer productivity. In *SIGSOFT '06/FSE-14: Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 1–11, New York, NY, USA, 2006. ACM Press.

[15] Gregor Kiczales, Jim des Rivières, and Daniel G. Bobrow. *The Art of the Metaobject Protocol*. MIT Press, 1991.

[16] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-Oriented Programming. In Mehmet Aksit and Satoshi Matsuoka, editors, *Proceedings ECOOP '97*, volume 1241 of *LNCS*, pages 220–242, Jyvaskyla, Finland, June 1997. Springer-Verlag.

[17] Jannik Laval, Alexandre Bergel, and Stéphane Ducasse. Matrice de dépendances enrichie. In *Proceedings of Languages et Modèles à Objets (LMO 2009)*, Nancy, France, 2009.

[18] Mariano Martinez Peck, Noury Bouraqadi, Marcus Denker, Stéphane Ducasse, and Luc Fabresse. Efficient proxies in Smalltalk. In *Proceedings of ESUG International Workshop on Smalltalk Technologies (IWST'11)*, Edinburgh, Scotland, 2011.

[19] Dehouck Mathieu, Usman Bhatti, Alexandre Bergel, and Stéphane Ducasse. Pragmatic visualizations for roassal: a florilegium. In *International Workshop on Smalltalk Technologies*, 2013.

[20] Sébastien Mosser, Alexandre Bergel, and Mireille Blay-Fornarino. Visualizing and assessing a compositional approach of business process design. In *Proceedings of 9th International Conference on Software Composition (SC'10)*. LNCS Springer Verlag, July 2010. to appear.

[21] Lukas Renggli, Stéphane Ducasse, Tudor Gîrba, and Oscar Nierstrasz. Practical dynamic grammars for dynamic languages. In *4th Workshop on Dynamic Languages and Applications (DYLA 2010)*, Malaga, Spain, June 2010.

[22] Romain Robbes, David Röthlisberger, and Éric Tanter. Extensions during software evolution: do objects meet their promise? In *Proceedings of the 26th European conference on Object-Oriented Programming*, ECOOP'12, pages 28–52, Berlin, Heidelberg, 2012. Springer-Verlag.

[23] Don Roberts, John Brant, Ralph E. Johnson, and Bill Opdyke. An automated refactoring tool. In *Proceedings of ICAST '96, Chicago, IL*, April 1996.

[24] David Röthlisberger, Marcus Denker, and Éric Tanter. Unanticipated partial behavioral reflection: Adapting applications at runtime. *Journal of Computer Languages, Systems and Structures*, 34(2-3):46–65, July 2008.

[25] Benjamin Van Ryseghem. Spec - technical report. Technical report, INRIA Lille Nord Europe, 2012.

[26] Benjamin Van Ryseghem, Stéphane Ducasse, and Johan 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)*, IWST '12, pages 2:1–2:14, Gent, Belgium, 2012. ACM.

[27] Juan Pablo Sandoval Alcocer, Alexandre Bergel, Stéphane Ducasse, and Marcus Denker. Performance evolution blueprint: Understanding the impact of software evolution on performance. In *Vissoft 2013*, 2013.

[28] Camille Teruel, Damien Cassou, and Stéphane Ducasse. Object Graph Isolation with Proxies. In *DYLA - 7th Workshop on Dynamic Languages and Applications, Collocated with 26th European Conference on Object-Oriented Programming - 2013*, 2013.