Final Scientific Report SNF Project no. 200020-131827
"Synchronizing Models and Code"

January 31, 2016

1 Summary of results

The goal of this project was to enable software developers to quickly and effectively analyze complex software systems with the help of tools to rapidly construct, query and manipulate software models. The key results achieved in each of the tracks of this project are as follows:

1. **Agile Modeling:** we developed and published a novel approach to imprecise parsing, called *Bounded Seas*.
2. **Meta-Tooling:** we developed several advanced prototypes of "moldable" developer tools that can be easily adapted to various application domains, and tested them extensively with professional developers.
3. **Large-Scale Software Analysis:** we have been exploring several parallel tracks on mining software ecosystems to support developer tasks and activities, such as lightweight type inference for dynamic languages, and fixes for frequently occurring bugs.
4. **Architectural Monitoring:** based on extensive empirical surveys, we have developed a high-level DSL for specifying and monitoring architectural constraints, and validated it with industrial case studies.

Results

The overall goals of the project have been described in a keynote paper presented at the International Conference on Program Comprehension [NL12].

Agile Modeling

In this track we have been exploring techniques to rapidly construct models from source code by using imprecise parsers. An overview paper [NK15] describes both the prior work and the research plan to explore island parsing, indentation-aware parsing and automatic keyword recognition as steps towards rapidly building imprecise parsers for a given language.

Island grammars are very promising, but they are very difficult to specify correctly, and they are very fragile to change, since the specification of the "water" to be ignored depends on the islands to be extracted. We have therefore been developing a new approach to island parsing called *bounded seas*, in which the water is effectively computed from the context of the islands to be recognized [KLN14a] [KLN14b] [KLIN15].

A prototype of bounded seas has been implemented in PetitParser, a framework for building composable parsers developed at the SCG [KLR$^+$13]. With the help of Masters and Bachelor students, we have been carrying out case studies with Java and Ruby to determine the effectiveness and robustness of bounded seas vs. traditional island parsers.

Since we also want to exploit structural clues in code to infer model elements, we have been extending PetitParser to support indentation-aware operators. Numerous languages, like Python, Haskell and F#, rely on indentation to define structure, but their indentation-aware features differ in significant ways. An early prototype was developed as a Bachelors project [Giv13].

We have also started with the help of another Bachelors student to explore the use of heuristics to automatically identify different classes of tokens, such as keywords, in unknown programming languages. Initial results are quite promising, and we plan to extend the set of heuristics in a followup project [Gug15].

Meta-Tooling

In this track we are investigating how to enable developers in rapidly adapting development tools to support specific application domains [CGN15].

To this end, we have produced a number of "moldable" developer tools. The Moldable Debugger [CNG13] [CGN14b] [CDGN15] is a debugging framework that can be easily adapted to different domains, such as event-driven systems, or parsing.

Another example of such a tool is the Moldable Inspector, a configurable object inspector that can easily be adapted to a given domain [CGN14a] [CGNS15a] [CGNS15b], and Spotter, a search tool that similarly adapts itself to various application domains [KBC^{+}15] [SCG^{+}15]. Domain-specific visualizations are a recurring theme in this work [GC15].

The moldable tools have been integrated into the latest version of the open-source Pharo development environment[1] and are used on a daily basis by professional developers.

In related work, we have been investigating how to offer very high-level support for analysis and software visualization of large software corpora [Mer14] [MLN15a] [MLN15b].

Finally, we have completed and published work started in the predecessor project on dynamic updates of running applications [WLN13] and supporting behavioural variations in running applications [WNTD14] [TWDN15].

Large-Scale Software Analysis

In this track we have been exploring numerous ways to exploit the large amount of data available in both the immediate ecosystem of a given application, and the broader context of open source software written in the same language.

In early work, we analysed the legacy PL/1 ecosystem of a large Swiss financial organization [ALNW13] [Aes13]. In other work initiated earlier but completed in the current project, we have developed novel techniques to efficiently detect software clones in very large corpora (*i.e.*, all available open-source Java code) [Vog14] [Sch14b]. These works have inspired many of the threads outlined below.

Pangea[2] is a workbench for analysing multi-language software corpora [CCSL14]. Pangea provides a parallel infrastructure together with a specialized DSL that eases the analysis of a large number of software models interpreted by the Moose analysis platform.

We have been exploring numerous applications of large-scale software analysis. By ranking most commonly accessed methods based on their popularity in the ecosystem, we can offer developers a more productive browsing experience that leads more quickly to relevant source code [SLN14] [SLN14b]. With Ecosystem-Aware Type Inference (EATI), we analyse the ecosystem of Smalltalk source code to determine to which concrete types given polymorphic variables are most commonly bound, and exploit this information back in the IDE [SLN14a]. We have analysed the prevalence of polymorphism in open-source Smalltalk and Java systems [MCL^{+}15], and we are investigating how to combine static and dynamic analyses to detect polymorphic usages cheaply and efficiently [Mil14]. Most recently, we exploit type hints in method argument names to effectively infer missing type information [SLN16].

We are also analysing software corpora to mine which kinds of bug fixes occur most frequently. Interestingly, in Java code the most common bug fix is to insert a "null-check" to ensure that a variable being accessed is properly initialised [OLN14]. An analysis of null checks in 810 open-source Java projects shows that 35% of all conditionals contain null checks and 71% of the value checked are return values

[1] http://pharo.org

[2] http://scg.unibe.ch/research/pangea

from method calls, suggesting interesting opportunities for improvements to the programming language and development environment to reduce the prevalence of such bugs [Osm15] [OLLN16].

In other related activity we have explored the impact of identifier names on program readability [LK13], tracked geographical knowledge transfer within StackOverflow [SL13], and developed a framework for hierarchical data analysis. [Sch14a]

Finally, we have carried out empirical studies with developers to better understand their needs with respect to the upstream (*i.e.*, providers) and downstream (*i.e.*, users) of ecosystem resources (*i.e.*, libraries and tools) [HLSN13] [HLSN14] [Hae14].

Architectural Monitoring

In this track we are exploring ways to track the evolution of a complex software system and monitor possible violations of architectural constraints.

In the conclusion of earlier work, we have developed tools and techniques to recover software architecture from an existing code base [LLN14] and we have developed techniques to predict dependencies in software systems using domain-based coupling [APL+14].

We carried out two extensive empirical studies of software practitioners to determine what kinds of architectural constraints are most important in practice. A first qualitative study identified which kinds of constraints and concerns arise in industrial projects in practice and a second, quantitative study measured the relative importance of these constraints [CLN14a]. An important outcome was to identify the areas practitioners consider critical and where tool support is lacking.

In subsequent work, we have developed *Dictō*, a high-level domain specific language for specifying and testing architectural rules [CLN14b] [CLN14c] [Car15b] [CLN15] [Car15a] [CLT+16]. Based on the results of the empirical studies, Dictō was designed to be able to easily express a large variety of architectural constraints. Dictō can be connected to various back-end tools to perform the actual analyses. Dictō has already been connected to a large variety of tools, and we have carried out several case studies with external partners to assess its effectiveness [Car16] [Tru15].

In related work, we have developed *Marea*, a tool to detect and break dependency cycles in software systems using an explicit profit and cost model to determine which dependencies are most "profitable" to break [Aga15] [CALN16].

Staff contributions

This project was supervised by Oscar Nierstrasz (Full Professor) and Mircea Lungu (Postdoc). Please note that Dr. Lungu has left Switzerland to assume a position as Assistant Professor (tenure-track) at the University of Groningen (Netherlands).

Here we summarize the contributions of the project staff (PhD students) throughout the project. Note that we only report on the staff whose salaries are paid from this project. In addition, Leonel Merino, Nevena Milojković and Haidar Osman have been contributing to this project, while their salaries have been paid by the University of Bern.

- Andrea Caracciolo has contributed to the Architectural Monitoring track, in which he carried out extensive qualitative and quantitative studies with software developers to determine how software architecture is specified and checked in practice [CLN14a]. He subsequently developed a high-level architectural monitoring language, Dictō [CLN14b] [CLN15] [CLN14c] [Car15b] [Car15a] [CLT+16], that offers a uniform interface to underlying tools to check constraints. An example is Marea [CALN16], a tool to detect and break dependency cycles. Andrea Caracciolo co-supervised two MSc theses related to this work [Tru15] [Aga15]. He is scheduled to defend his PhD thesis [Car16] on March 22, 2016.

- Andrei Chiş has been working on the Meta-Tooling track on adaptable, or "moldable" development tools [CGN15]. He received the Best Student Paper award for his full paper on the Moldable Debugger at the Software Language Engineering (SLE) conference 2014 [CGN14b]. An extended journal paper has also been published on the Moldable Debugger [CDGN15]. Andrei Chiş received a European Smalltalk User Group 2014 Technology Innovation Award (1st prize) for his implementation of the Moldable Inspector [CGNS15a] [CGNS15b]. He also received a SPLASH 2015 Distinguished Demo Award for presenting the Moldable Inspector [CGN14a] [GC15], and a European Smalltalk User Group 2015 Technology Innovation Award (1st prize) for the Spotter search tool [SCG$^+$15] [KBC$^+$15]. Andrei Chiş is scheduled to defend his thesis in September 2016.

- Jan Kurš has been working on the Agile Modeling track [NK15]. He developed *Bounded Seas*, a novel approach to imprecise parsing [KLN14a] [KLN14b], and has published a journal paper on the approach [KLIN15]. The approach has been validated in the PetitParser parsing framework [KLR$^+$13]. Jan Kurš has co-supervised two Bachelors theses on related themes [Giv13] [Gug15] and a further MSc thesis (in progress). He is currently on a four month internship at Google (Mountain View CA), and plans to defend his thesis in the Spring of 2016.

- Nikolaus Schwarz was mainly employed on the predecessor project (Synchronizing Models and Code, 200020_131827), and worked for only one month on this project. He successfully defended his PhD thesis, entitled *Scaleable Code Clone Detection* [Sch14b], in February 2014. Since then he has been employed as a researcher at Google (Zürich).

- Boris Spasojević has been working on the Large-Scale Software Analysis track, specifically on ecosystem mining tools [CCSL14], on mining the software ecosystem to improve type inference [SLN14a] [SLN16], and on helping developers to rapidly identify relevant source code [SLN14] [SLN14b]. He spent three months at Google (Munich) on an internship in the summer of 2015. Boris Spasojević is scheduled to defend his PhD at the end of 2016 or early 2017.

2 Research output

All reported publications are available electronically from the project's home page:

`http://scg.unibe.ch/asa`

We also list selected Bachelors and Masters theses directly relevant to this project.

Journal papers

[APL+14] Amir Aryani, Fabrizio Perin, Mircea Lungu, Abdun Naser Mahmood, and Oscar Nierstrasz. Predicting dependencies using domain-based coupling. *Journal of Software: Evolution and Process*, 26(1):50–76, 2014. URL: `http://scg.unibe.ch/archive/papers/Arya14aJSME.pdf`, `doi:10.1002/smr.1598`.

[CDGN15] Andrei Chiş, Marcus Denker, Tudor Gîrba, and Oscar Nierstrasz. Practical domain-specific debuggers using the moldable debugger framework. *Computer Languages, Systems & Structures*, 44, Part A:89–113, 2015. Special issue on the 6th and 7th International Conference on Software Language Engineering (SLE 2013 and SLE 2014). URL: `http://scg.unibe.ch/archive/papers/Chis15c-PracticalDomainSpecificDebuggers.pdf`, `doi:10.1016/j.cl.2015.08.005`.

[KLIN15] Jan Kurš, Mircea Lungu, Rathesan Iyadurai, and Oscar Nierstrasz. Bounded seas. *Computer Languages, Systems & Structures*, 44, Part A:114 – 140, 2015. Special issue on the 6th and 7th International Conference on Software Language Engineering (SLE 2013 and SLE 2014). URL: `http://scg.unibe.ch/archive/papers/Kurs15a-BoundedSeas.pdf`, `doi:10.1016/j.cl.2015.08.002`.

[LLN14] Mircea Lungu, Michele Lanza, and Oscar Nierstrasz. Evolutionary and collaborative software architecture recovery with Softwarenaut. *Science of Computer Programming*, 79(0):204 – 223, 2014. URL: `http://scg.unibe.ch/archive/papers/Lung14a.pdf`, `doi:10.1016/j.scico.2012.04.007`.

[NK15] Oscar Nierstrasz and Jan Kurš. Parsing for agile modeling. *Science of Computer Programming*, 97, Part 1(0):150–156, 2015. URL: `http://scg.unibe.ch/archive/papers/Nier13cAgileModeling.pdf`, `doi:10.1016/j.scico.2013.11.011`.

[TWDN15] Camille Teruel, Erwann Wernli, Stéphane Ducasse, and Oscar Nierstrasz. Propagation of behavioral variations with delegation proxies. *Transactions on Aspect-Oriented Software Development XII*, 8989:63–95, 2015. URL: `http://scg.unibe.ch/archive/papers/Teru15a-delegation-proxies.pdf`, `doi:10.1007/978-3-662-46734-3_2`.

[WLN13] Erwann Wernli, Mircea Lungu, and Oscar Nierstrasz. Incremental dynamic updates with first-class contexts. *Journal of Object Technology*, 12(3):1:1–27, August 2013. URL: `http://scg.unibe.ch/archive/papers/Wern13a.pdf`, `doi:10.5381/jot.2013.12.3.a1`.

Conference papers

[ALNW13] Erik Aeschlimann, Mircea Lungu, Oscar Nierstrasz, and Carl Worms. Analyzing PL/1 legacy ecosystems: An experience report. In *Proceedings of the 20th Working Conference on Reverse Engineering, WCRE 2013*, pages 441 – 448, 2013. URL: `http://scg.unibe.ch/archive/papers/Aesc13a-PL1Ecosystem.pdf`, `doi:10.1109/WCRE.2013.6671320`.

[CALN16] Andrea Caracciolo, Bledar Aga, Mircea Lungu, and Oscar Nierstrasz. Marea: a semi-automatic decision support system for breaking dependency cycles. In *Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, March 2016. to appear. URL: `http://scg.unibe.ch/archive/papers/Cara16b.pdf`.

[CCSL14] Andrea Caracciolo, Andrei Chiş, Boris Spasojević, and Mircea Lungu. Pangea: A workbench for statically analyzing multi-language software corpora. In *Source Code Analysis and Manipulation (SCAM), 2014 IEEE 14th International Working Conference on*, pages 71–76. IEEE, September 2014. URL: `http://scg.unibe.ch/archive/papers/Cara14c.pdf`, `doi:10.1109/SCAM.2014.38`.

[CGN14b] Andrei Chiş, Tudor Gîrba, and Oscar Nierstrasz. The Moldable Debugger: A framework for developing domain-specific debuggers. In Benoît Combemale, DavidJ. Pearce, Olivier Barais, and Jurgen J. Vinju, editors, *Software Language Engineering*, volume 8706 of *Lecture Notes in Computer Science*, pages 102–121. Springer International Publishing, 2014. URL: `http://scg.unibe.ch/archive/papers/Chis14b-MoldableDebugger.pdf`, `doi:10.1007/978-3-319-11245-9_6`.

[CGNS15a] Andrei Chiş, Tudor Gîrba, Oscar Nierstrasz, and Aliaksei Syrel. The Moldable Inspector. In *Proceedings of the 2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! 2015, pages 44–60, New York, NY, USA, 2015. ACM. URL: `http://scg.unibe.ch/archive/papers/Chis15a-MoldableInspector.pdf`, `doi:10.1145/2814228.2814234`.

[CLN14a] Andrea Caracciolo, Mircea Lungu, and Oscar Nierstrasz. How do software architects specify and validate quality requirements? In *European Conference on Software Architecture (ECSA)*, volume 8627 of *Lecture Notes in Computer Science*, pages 374–389. Springer Berlin Heidelberg, August 2014. URL: `http://`

scg.unibe.ch/archive/papers/Cara14a-SpecifyValidateQualityRequirements.pdf, doi:10.1007/978-3-319-09970-5_32.

[CLN15] Andrea Caracciolo, Mircea Lungu, and Oscar Nierstrasz. A unified approach to architecture conformance checking. In *Proceedings of the 12th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, pages 41–50. ACM Press, May 2015. URL: http://scg.unibe.ch/archive/papers/Cara15b.pdf, doi:10.1109/WICSA.2015.11.

[GC15] Tudor Gîrba and Andrei Chiş. Pervasive Software Visualizations. In *Proceedings of 3rd IEEE Working Conference on Software Visualization*, VISSOFT'15, pages 1–5. IEEE, September 2015. URL: http://scg.unibe.ch/archive/papers/Girb15b-PervasiveSoftwareVisualizations.pdf, doi:10.1109/VISSOFT.2015.7332409.

[KBC$^+$15] Juraj Kubelka, Alexandre Bergel, Andrei Chiş, Tudor Gîrba, Stefan Reichhart, Romain Robbes, and Aliaksei Syrel. On understanding how developers use the Spotter search tool. In *Proceedings of 3rd IEEE Working Conference on Software Visualization - New Ideas and Emerging Results*, VISSOFT-NIER'15, pages 145–149. IEEE, September 2015. URL: http://scg.unibe.ch/archive/papers/Kube15a-OnUnderstandingHowDevelopersUseTheSpotterSearchTool.pdf, doi:10.1109/VISSOFT.2015.7332426.

[KLN14b] Jan Kurš, Mircea Lungu, and Oscar Nierstrasz. Bounded seas: Island parsing without shipwrecks. In Benoît Combemale, David J. Pearce, Olivier Barais, and Jurgen J. Vinju, editors, *Software Language Engineering*, volume 8706 of *Lecture Notes in Computer Science*, pages 62–81. Springer International Publishing, 2014. URL: http://scg.unibe.ch/archive/papers/Kurs14b-BoundedSeas.pdf, doi:10.1007/978-3-319-11245-9_4.

[MCL$^+$15] Nevena Milojković, Andrea Caracciolo, Mircea Lungu, Oscar Nierstrasz, David Röthlisberger, and Romain Robbes. Polymorphism in the spotlight: Studying its prevalence in Java and Smalltalk. In *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*, pages 186–195. IEEE Press, 2015. Published. URL: http://scg.unibe.ch/archive/papers/Milo15a.pdf, doi:10.1109/ICPC.2015.29.

[MLN15b] Leonel Merino, Mircea Lungu, and Oscar Nierstrasz. Explora: A visualisation tool for metric analysis of software corpora. In *VISSOFT'15: Proceedings of the 3rd IEEE Working Conference on Software Visualization*, pages 195–199. IEEE, 2015. URL: http://scg.unibe.ch/archive/papers/Meri15b.pdf, doi:10.1109/VISSOFT.2015.7332436.

[NL12] Oscar Nierstrasz and Mircea Lungu. Agile software assessment. In *Proceedings of International Conference on Program Comprehension (ICPC 2012)*, pages 3–10, 2012. URL: http://scg.unibe.ch/archive/papers/Nier12bASA.pdf, doi:10.1109/ICPC.2012.6240507.

[OLLN16] Haidar Osman, Manuel Leuenberger, Mircea Lungu, and Oscar Nierstrasz. Tracking null checks in open-source Java systems. In *Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, March 2016. to appear. URL: http://scg.unibe.ch/archive/papers/Osma16a.pdf.

[OLN14] Haidar Osman, Mircea Lungu, and Oscar Nierstrasz. Mining frequent bug-fix code changes. In *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on*, pages 343–347, February 2014. URL: http://scg.unibe.ch/archive/papers/Osma14aMiningBugFixChanges.pdf, doi:10.1109/CSMR-WCRE.2014.6747191.

[SCG$^+$15] Aliaksei Syrel, Andrei Chiş, Tudor Gîrba, Juraj Kubelka, Oscar Nierstrasz, and Stefan Reichhart. Spotter: towards a unified search interface in IDEs. In *Proceedings of the Companion Publication of the 2015 ACM SIGPLAN Conference on Systems, Programming, and Applications: Software for Humanity*, SPLASH Companion 2015, pages 54–55, New York, NY, USA, 2015. ACM. URL: http://scg.unibe.ch/archive/papers/Syre15a-SpotterPosterAbstract.pdf, doi:10.1145/2814189.2817269.

[SLN14a] Boris Spasojević, Mircea Lungu, and Oscar Nierstrasz. Mining the ecosystem to improve type inference for dynamically typed languages. In *Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! '14, pages 133–142, New York, NY, USA, 2014. ACM. URL: http://scg.unibe.ch/archive/papers/Spas14c.pdf, doi:10.1145/2661136.2661141.

[SLN14b] Boris Spasojević, Mircea Lungu, and Oscar Nierstrasz. Overthrowing the tyranny of alphabetical ordering in documentation systems. In *2014 IEEE International Conference on Software Maintenance and Evolution (ERA Track)*, pages 511–515, September 2014. URL: http://scg.unibe.ch/archive/papers/Spas14b.pdf, doi:10.1109/ICSME.2014.84.

[SLN16] Boris Spasojević, Mircea Lungu, and Oscar Nierstrasz. A case study on type hints in method argument names in Pharo Smalltalk projects. In *Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, March 2016. to appear. URL: http://scg.unibe.ch/archive/papers/Spas16a.pdf.

[WNTD14] Erwann Wernli, Oscar Nierstrasz, Camille Teruel, and Stephane Ducasse. Delegation proxies: The power of propagation. In *Proceedings of the 13th International Conference on Modularity*, MODULARITY '14, pages 1–12, New York, NY, USA, 2014. ACM. URL: http://scg.unibe.ch/archive/papers/Wern14a.pdf, doi:10.1145/2577080.2577081.

International Workshop papers

[Car15a] Andrea Caracciolo. On the evaluation of a DSL for architectural consistency checking. In *Extended Abstracts of the Eighth Seminar on Advanced Techniques and Tools for Software Evolution (SATToSE 2015)*, pages 55–57, July 2015. URL: http://scg.unibe.ch/archive/papers/Cara15c.pdf.

[Car15b] Andrea Caracciolo. A unified approach to automatic testing of architectural constraints. In *Proceedings of ICSE 2015 (37st International Conference on Software Engineering), Doctoral Symposium*, volume 2, pages 871–874. ACM Press, 2015. URL: http://scg.unibe.ch/archive/papers/Cara15a.pdf, doi:10.1109/ICSE.2015.281.

[CGN14a] Andrei Chiş, Tudor Gîrba, and Oscar Nierstrasz. The Moldable Inspector: a framework for domain-specific object inspection. In *Proceedings of International Workshop on Smalltalk Technologies (IWST 2014)*, 2014. URL: http://scg.unibe.ch/archive/papers/Chis14a-MoldableInspector.pdf.

[CGN15] Andrei Chiş, Tudor Gîrba, and Oscar Nierstrasz. Towards moldable development tools. In *Proceedings of the 6th Workshop on Evaluation and Usability of Programming Languages and Tools*, PLATEAU '15, 2015. URL: http://scg.unibe.ch/archive/papers/Chis15d_TowardsMoldableDevelopmentTools.pdf, doi:10.1145/2846680.2846684.

[CGNS15b] Andrei Chiş, Tudor Gîrba, Oscar Nierstrasz, and Aliaksei Syrel. GTInspector: A moldable domain-aware object inspector. In *Proceedings of the Companion Publication of the 2015 ACM SIGPLAN Conference on Systems, Programming, and Applications: Software for Humanity*, SPLASH Companion 2015, pages 15–16, New York, NY, USA, 2015. ACM. URL: http://scg.unibe.ch/archive/papers/Chis15b-GTInspector.pdf, doi:10.1145/2814189.2814194.

[CLN14b] Andrea Caracciolo, Mircea Lungu, and Oscar Nierstrasz. Dicto: A unified DSL for testing architectural rules. In *Proceedings of the 2014 European Conference on Software Architecture Workshops*, ECSAW '14, pages 21:1–21:4, New York, NY, USA, 2014. ACM. URL: http://scg.unibe.ch/archive/papers/Cara14b-Dicto.pdf, doi:10.1145/2642803.2642824.

[CLT$^+$16] Andrea Caracciolo, Mircea Lungu, Oskar Truffer, Kirill Levitin, and Oscar Nierstrasz. Evaluating an architecture conformance monitoring solution. In *Proceedings of the 7th IEEE International Workshop on Empirical Software Engineering in Practice (IWESEP)*, March 2016. to appear. URL: http://scg.unibe.ch/archive/papers/Cara16c.pdf.

[CNG13] Andrei Chiş, Oscar Nierstrasz, and Tudor Gîrba. Towards a moldable debugger. In *Proceedings of the 7th Workshop on Dynamic Languages and Applications*, 2013. URL: http://scg.unibe.ch/archive/papers/Chis13a-TowardsMoldableDebugger.pdf, doi:10.1145/2489798.2489801.

[HLSN13] Nicole Haenni, Mircea Lungu, Niko Schwarz, and Oscar Nierstrasz. Categorizing developer information needs in software ecosystems. In *Proceedings of the 1st Workshop on Ecosystem Architectures*, pages 1–5, 2013. URL: http://scg.unibe.ch/archive/papers/Haen13a-EcosystemInformationNeeds.pdf.

[HLSN14] Nicole Haenni, Mircea Lungu, Niko Schwarz, and Oscar Nierstrasz. A quantitative analysis of developer information needs in software ecosystems. In *Proceedings of the 2nd Workshop on Ecosystem Architectures (WEA'14)*, pages 1–6, 2014. URL: http://scg.unibe.ch/archive/papers/Haen14a-QuantitativeEcosystemInformationNeeds.pdf, doi:10.1145/2642803.2642815.

[KLN14a] Jan Kurš, Mircea Lungu, and Oscar Nierstrasz. Top-down parsing with parsing contexts. In *Proceedings of International Workshop on Smalltalk Technologies (IWST 2014)*, 2014. URL: http://scg.unibe.ch/archive/papers/Kurs14a-ParsingContext.pdf.

[LK13] Mircea Lungu and Jan Kurš. On planning an evaluation of the impact of identifier names on the readability and maintainability of programs. In *USER'13: Proceedings of the 2nd Workshop on User evaluations for Software Engineering Researchers*, pages 13 – 15, 2013. URL: http://scg.unibe.ch/archive/papers/Lung13a-Planning.pdf.

[Mer14] Leonel Merino. Adaptable visualisation based on user needs. In *SATToSE'14: Pre-Proceedings of the 7th International Seminar Series on Advanced Techniques & Tools for Software Evolution*, pages 71–74, July 2014. URL: http://scg.unibe.ch/archive/papers/Meri14a.pdf.

[Mil14] Nevena Milojković. Towards cheap, accurate polymorphism detection. In *SATToSE'14: Pre-Proceedings of the 7th International Seminar Series on Advanced Techniques & Tools for Software Evolution*, pages 54–55, July 2014. URL: http://scg.unibe.ch/archive/papers/Milo14a.pdf.

[MLN15a] Leonel Merino, Mircea Lungu, and Oscar Nierstrasz. Explora: Infrastructure for scaling up software visualisation to corpora. In *SATToSE'14: Post-Proceedings of the 7th International Seminar Series on Advanced Techniques & Tools for Software Evolution*, volume 1354. CEUR Workshop Proceedings (CEUR-WS.org), 2015. http://ceur-ws.org/Vol-1354/. URL: http://scg.unibe.ch/archive/papers/Meri15a.pdf.

[Osm15] Haidar Osman. Null check analysis. In *Extended Abstracts of the Eighth Seminar on Advanced Techniques and Tools for Software Evolution (SATToSE 2015)*, pages 86–88, July 2015. URL: http://scg.unibe.ch/archive/papers/Osma15a.pdf.

[SL13] Dennis Schenk and Mircea Lungu. Geo-locating the knowledge transfer in stackoverflow. In *Proceedings of the 5th International Workshop on Social Software Engineering*, pages 21–24, 2013. URL: http://scg.unibe.ch/archive/papers/Sche13a-GeolocatingStackOverflow.pdf.

[SLN14] Boris Spasojević, Mircea Lungu, and Oscar Nierstrasz. Towards faster method search through static ecosystem analysis. In *Proceedings of the 2014 European Conference on Software Architecture Workshops*, ECSAW '14, pages 11:1–11:6, New York, NY, USA, August 2014. ACM. URL: `http://scg.unibe.ch/archive/papers/Spas14aFasterMethodLookup.pdf`, `doi:10.1145/2642803.2642814`.

Book chapters

[KLR+13] Jan Kurš, Guillaume Larcheveque, Lukas Renggli, Alexandre Bergel, Damien Cassou, Stéphane Ducasse, and Jannik Laval. PetitParser: Building modular parsers. In *Deep Into Pharo*, page 36. Square Bracket Associates, September 2013. URL: `http://scg.unibe.ch/archive/papers/Kurs13a-PetitParser.pdf`.

Miscellaneous

[CLN14c] Andrea Caracciolo, Mircea Lungu, and Oscar Nierstrasz. Dicto: Keeping software architecture under control. *ERCIM News*, 99, October 2014. URL: `http://ercim-news.ercim.eu/en99/special/dicto-keeping-software-architecture-under-control`.

Selected PhD, MSc and Bachelors Theses

[Aes13] Erik Aeschlimann. St1-PL/1: Extracting quality information from PL/1 legacy ecosystems. Masters thesis, University of Bern, December 2013. URL: `http://scg.unibe.ch/archive/masters/Aesc13b.pdf`.

[Aga15] Bledar Aga. Marea — a tool for breaking dependency cycles between packages. Masters thesis, University of Bern, January 2015. URL: `http://scg.unibe.ch/archive/masters/Aga15a.pdf`.

[Car16] Andrea Caracciolo. *A Unified Approach to Architecture Conformance Checking*. PhD thesis, University of Bern, March 2016. URL: `http://scg.unibe.ch/archive/phd/caracciolo-phd.pdf`.

[Giv13] Attieh Sadeghi Givi. Layout sensitive parsing in the PetitParser framework. Bachelor's thesis, University of Bern, October 2013. URL: `http://scg.unibe.ch/archive/projects/Sade13a.pdf`.

[Gug15] Joël Guggisberg. Automatic token classification — an attempt to mine useful information for parsing. Bachelor's thesis, University of Bern, December 2015. URL: `http://scg.unibe.ch/archive/projects/Gugg15a.pdf`.

[Hae14] Nicole Haenni. Information needs in software ecosystems development — a contribution to improve tool support across software systems. Masters thesis, University of Bern, September 2014. URL: `http://scg.unibe.ch/archive/masters/Haen14b.pdf`.

[Sch14a] Dennis Schenk. Quicksilver — a framework for hierarchical data analysis. Masters thesis, University of Bern, September 2014. URL: `http://scg.unibe.ch/archive/masters/Sche14a.pdf`.

[Sch14b] Niko Schwarz. *Scaleable Code Clone Detection*. PhD thesis, University of Bern, February 2014. URL: `http://scg.unibe.ch/archive/phd/schwarz-phd.pdf`.

[Tru15] Oskar Truffer. Continuous integration with architectural invariants — a case study about architectural monitoring in practice. Masters thesis, University of Bern, December 2015. URL: `http://scg.unibe.ch/archive/masters/Truf15a.pdf`.

[Vog14] Simon Vogt. Clone detection that scales. Masters thesis, University of Bern, February 2014. URL: `http://scg.unibe.ch/archive/masters/Vogt14a.pdf`.