Final Scientific Report
SNF Project no. 200020-121594
"Bringing Models Closer to Code"

November 11, 2010

a) Summary of results

This project explores various ways of synchronizing software source code with implicit application domain knowledge. The key results achieved in the four tracks of this project include:

1. **Coordinating models and code:** We have developed a technique called *live feature analysis* for establishing the correspondence between a feature and the code that implements it. Live feature analysis allows us to maintain at runtime a model of a system's features. To enable change at runtime, a system must be self-aware and be able to fully reflect on itself. To achieve this we developed *Albedo*, a unified approach to structural and behavioral reflection. We have also explored how search-driven-development can improve the understanding of the code by raising the trustability of the search results.

2. **Embedding domain models in the code:** We continued the development of the *Helvetia* platform for embedding domain-specific languages into an existing programming environment by leveraging the underlying representation of the host language and the existing tools. We have extended Helvetia with *Language Boxes*, which allow language extensions to be scoped to a particular context, and we have developed *PetitParser*, which enables multiple grammar extensions to be easily composed.

3. **Bringing dynamic models to the IDE:** We have extended the developed approaches integrating dynamic models into the IDE, have empirically analyzed the problems of navigating software systems, and conducted controlled experiments with professional software developers to reveal the effect of the availability of dynamic models in the IDE on developer productivity. These experiments showed that developers can both more quickly and more correctly perform typical software development and maintenance tasks thanks to the access to dynamic models in the IDE.

4. **Model-centric development:** We have developed a new Smalltalk environment called *Pinocchio* in which the application code and interpreter are reified. It allows us to extend and modify the interpreter from within the provided runtime by subclassing its base classes. During the last year we focused on finishing the compiler and runtime so that it can run standard Smalltalk code with reasonable performance. To do so we switched to an opcode-based runtime leaving the code reified as AST nodes only for the first-class interpreters. Then we used the infrastructure to prototype three non-trivial interpreters: a stepping debugger, a parallel debugger and an alias interpreter.

This research of the final year has resulted in 2 journal papers, 8 papers in international, peer-reviewed conferences, as well as numerous other workshop papers, reports and theses.

Results

We present the results obtained during the period from Oct 1, 2009 to Sept 30, 2010.

Coordinating models and code

In this track we explore ways to embed multiple models in code and to mine higher-level abstractions from code. We explored how search-driven-development can improve the understanding of the code by enabling the developer to incrementally construct his mental model of the source code. We investigated the importance of search-driven navigation in the IDE [Kuh10]. Furthermore we proposed a technique for raising the trustability of the search results by incorporating user votes and cross-project developer activity [GK10].

Traditional feature analysis approaches establish the correspondence between features and source code by exercising features to generate a trace of runtime events. This trace is then processed by post-mortem analysis. These approaches are neither dynamic nor adaptable to changes in the analyzed applications. To overcome these drawbacks we developed a new approach called *live feature analysis* [DRGN10]. This approach builds the causal connection between features and source code at runtime by adapting the application's behavior and annotating the structural representation of the source code. Our approach enables analysis of features on a running system, makes it possible to "grow" feature representations by exercising different scenarios of the same feature, and identifies execution elements down to the sub-method level

Runtime is of the utmost importance for software evolution. To enable change at runtime, a system must be self-aware and be able to fully reflect on itself. Most languages provide incomplete reflective facilities generally as external mechanisms. We developed Albedo [RRGN10] to face these challenges. Albedo is a model of fined-grained unanticipated dynamic structural and behavioral adaptation. Instead of providing reflective capabilities as an external mechanism we integrate them deeply in the environment. Modeling explicit meta-objects allows us to provide a range of reflective features and thereby evolve both application models and environments at run-time.

Concurrency control in most programming languages is expressed at a low level of abstraction, and is tangled with functional code. We introduced a novel concurrency control approach which uses high-level synchronization specifications to declaratively express concurrency requirements [RN09]. A dynamic synchronization system adapts functional code accordingly to meet those requirements.

Embedding domain models in the code

With *Helvetia* we explored a lightweight approach to leverage the underlying representation of the host language to embed new languages into an existing host environment. Helvetia is an extensible system that intercepts the compilation pipeline of the host language and reuses various tools such as editors and debuggers to seamlessly integrate language extensions [RGN10].

On top of Helvetia we proposed the *Language Boxes* model, a modular mechanism to encapsulate (1) compositional changes to the host language, (2) transformations to address various concerns such as compilation and syntax highlighting, and (3) scoping rules to control visibility of fine-grained language extensions. Language boxes enable multiple context-dependent language extensions [RDN09]. We then developed *PetitParser*, a dynamic grammar description framework. PetitParser makes it possible to dynamically transform, reuse, compose and extend language grammars; it is the enabling technology for Language Boxes [RDGN10b].

In extended case studies we have implemented various embedded languages[1] and used them as a validation of our approach. We have applied the Helvetia infrastructure to detect common problems in domain-specific code and display and fix these problems using the existing infrastructure [RDGN10a], and we have used Helvetia to integrate fine-grained, context-dependent models with application code.

Bringing dynamic models to the IDE

This track is concerned with the integration of run-time information into the static views of source code typically offered by Integrated Development Environments (IDEs). Empirical validation is a key component of this task.

[1] `http://scg.unibe.ch/research/helvetia/examples`

We have developed *Senseo*[2], a plug-in to the popular Eclipse IDE for Java [Röt10]. Senseo analyzes the runtime of Java applications to gather information about message sends or types of objects instantiated at runtime. To perform the dynamic analysis Senseo uses a dedicated Java instrumentation framework developed by the research group of Prof. Walter Binder at the University of Lugano. Senseo augments the Eclipse IDE with dynamic information by embedding information about dynamic callers or callees of a method in the source code view, by integrating easy-to-read visualizations pinpointing hot spots in code such as methods instantiating many large objects, or by listing all source entities collaborating to a selected artifact (package, class, or method).

An extensive empirical study with a large group of professional developers from industry has been carried out to assess the impact of Senseo on productivity for typical maintenance tasks. We have measured the time developers spent on the tasks and the correctness of the solutions. The study has revealed a 17.5% decrease in time spent and a 33.5% increase in correctness of the solutions when dynamic information is made available with Senseo in the Eclipse IDE [RHV+10] (in print).

As developers are overwhelmed with too much information in an IDE, we have developed *Autumn-Leaves*, a technique to automatically close unused windows or tabs in an IDE [RND09]. This approach helps developers to maintain the overview when working on large software systems whose navigation typically involves the opening of plenty of windows in the IDE.

Model-centric development

To support development tools like debuggers, software systems need to provide a meta-programming interface to alter their semantics and access internal data. Reflective capabilities are typically fixed at the level of the Virtual Machine (VM). Unanticipated reflective features must either be simulated by complex program transformations, or they require the development of a specially tailored VM.

To overcome this limitation we developed new Smalltalk environment called *Pinocchio* [VBG+10] [3]. It eliminates the restrictions on the reflective capabilities by fully providing them from within the runtime. To do so Pinocchio reifies the application code in the form of AST nodes and provides an extensible meta-circular interpreter for these AST nodes.

By providing a Smalltalk-based implementation of the code and its interpretation, Smalltalk programmers regain full runtime control over their code and its interpretation without having to rely on knowledge of a low-level language. To implement an alternative interpretation of an application the user can rely on the full power of Smalltalk to modify and extend the standard interpretation by subclassing the provided meta-circular interpreter.

We validated Pinocchio by implementing non-trivial examples that extend runtime semantics to support debugging, parallel debugging, and back-in-time object-flow debugging.

Since the implementation is fully decoupled from direct VM support we are confident that this strategy is a viable strategy for building custom interpreters and debuggers for other languages and environments.

Staff contributions

- Lukas Renggli has defended his PhD in October 2010 [Ren10]. He has developed a novel approach to integrate embedded languages within a general purpose programming language and the existing tools. He has participated in numerous papers related to his research, either as first author or as co-author [RG09, RDN09, RRGN10, PRR10, RGN10, RDGN10a, RDGN10b].

- Jorge Ressia has completed the second year of his PhD. He is working on the track "Coordinating models and code", and has developed a unified approach to achieve structural and behavioral reflection. He has published some initial results related to this work [RN09, RRGN10, DRGN10].

- David Röthlisberger completed his PhD at the beginning of June. He has extensively worked on techniques to augment the understanding of software systems from within the IDE, for instance by embedding dynamic information into the static source perspectives of IDEs. He validated the impact

[2] `http://scg.unibe.ch/research/senseo`
[3] `http://scg.unibe.ch/research/pinocchio`

of these techniques on developer productivity by conducting comprehensive empirical experiments with industrial software developers.

Dr. Röthlisberger has published about these techniques at various venues [RNDB09, RND09, Röt09, RHV+10, Röt10]. Dr. Röthlisberger also co-organized the PCODA workshop co-located with WCRE 2010 [HLRZG10].

- Toon Verwaest has completed the third year of his PhD. He has developed Pinocchio, a novel Smalltalk implementation that provides a high-level model of the application code as well as an extensible meta-circular interpreter to support continuous behavioral reflection. He has published the results of this work [VBG+10].

Important events

- Oscar Nierstrasz was co-organizer of FOSD 20010 (2nd Workshop on Feature-Oriented Software Development) and Invited Speaker at SOFSEM 2010 (36th International Conference on Current Trends in Theory and Practice of Computer Science, Principles of Software Construction track)
- Oscar Nierstrasz was program committee member of the international conferences TOOLS 2010, ICSM 2010, ETSM 2009, and the international workshops FAMOOSr 2009, FOSD 2009, IWST09, RAM-SE 2009, PCODA 2010, SEAA-EDISON 2010, RAM-SE 2010, Suite 2010, WETSoM 2010, and OOPS 2010.
- Lukas Renggli was a program committee member of the IDM 2009 conference, the IWST 2009 international workshop, the DSAL 2010 workshop, the IWST 2010 international workshop, and the FAST 2010 research track.
- Jorge Ressia has been co-reviewer of the international conferences MODELS 2009, ICSM 2010 and TOOLS 2010, and the FAST 2010 research track.
- David Röthlisberger co-organized the 5th PCODA international workshop on Program Comprehension through Dynamic Analysis (Boston, USA) [HLRZG10]
- Toon Verwaest was co-reviewer for TOOLS 2010, SE 2011 and C5 2011.

b) Publications

Papers published in the final year of this project are annexed to this report. They are all available electronically as PDF files at the following url:

`http://scg.unibe.ch/scgbib?query=snf10`

Please note that proceedings, theses, technical reports and student projects are *not* included with this report, but are nevertheless available electronically from the above URL.

Published papers

[ADGN10] Gabriela Arévalo, Stéphane Ducasse, Silvia Gordillo, and Oscar Nierstrasz. Generating a catalog of unanticipated schemas in class hierarchies using formal concept analysis. *Information and Software Technology*, In Press, Corrected Proof(52):1167–1187, December 2010.

[DDL09] Stéphane Ducasse, Marcus Denker, and Adrian Lienhard. Evolving a reflective language. In *Proceedings of the International Workshop on Smalltalk Technologies (IWST'09)*, pages 82–86, New York, NY, USA, jun 2009. ACM.

[DRGN10] Marcus Denker, Jorge Ressia, Orla Greevy, and Oscar Nierstrasz. Modeling features at runtime. In *Proceedings of MODELS 2010 Part II*, volume 6395 of *LNCS*, pages 138–152. Springer-Verlag, October 2010.

[GK10] Florian S. Gysin and Adrian Kuhn. A trustability metric for code search based on developer karma. In *ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation, 2010. SUITE '10.*, 2010.

[Gys10] Florian S. Gysin. Improved social trustability of code search results. In *Proceedings International Conference on Software Engineering, ICSE '10, Student Research Competition*, 2010.

[HLRZG10] Abdelwahab Hamou-Lhadj, David Röthlisberger, Andy Zaidman, and Orla Greevy. Workshop on program comprehension through dynamic analysis (PCODA). In *Proceedings of IEEE 17th Working Conference on Software Maintenance and Reengineering (WCRE)*, October 2010. To appear.

[KELN10] Adrian Kuhn, David Erni, Peter Loretan, and Oscar Nierstrasz. Software cartography: Thematic software visualization with consistent layout. *Journal of Software Maintenance and Evolution (JSME)*, 22(3):191–210, April 2010.

[Kuh10] Adrian Kuhn. Immediate search in the ide as an example of socio-technical congruence in search-driven development. In *ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation, 2010. SUITE '10.*, 2010.

[NG10] Oscar Nierstrasz and Tudor Gîrba. Lessons in software evolution learned by listening to Smalltalk. In J. van Leeuwen et al., editor, *SOFSEM 2010*, volume 5901 of *LNCS*, pages 77–95. Springer-Verlag, 2010.

[PRR10] Fabrizio Perin, Lukas Renggli, and Jorge Ressia. Ranking software artifacts. In *4th Workshop on FAMIX and Moose in Reengineering (FAMOOSr 2010)*, 2010.

[RDGN10a] Lukas Renggli, Stéphane Ducasse, Tudor Gîrba, and Oscar Nierstrasz. Domain-specific program checking. In Jan Vitek, editor, *Proceedings of the 48th International Conference on Objects, Models, Components and Patterns (TOOLS'10)*, volume 6141 of *LNCS*, pages 213–232. Springer-Verlag, 2010.

[RDGN10b] 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.

[RDN09] Lukas Renggli, Marcus Denker, and Oscar Nierstrasz. Language Boxes: Bending the host language with modular language changes. In *Software Language Engineering: Second International Conference, SLE 2009, Denver, Colorado, October 5-6, 2009*, volume 5969 of *LNCS*, pages 274–293. Springer, 2009.

[RG09] Lukas Renggli and Tudor Gîrba. Why Smalltalk wins the host languages shootout. In *Proceedings of International Workshop on Smalltalk Technologies (IWST 2009)*, pages 107–113, New York, NY, USA, 2009. ACM.

[RGN10] Lukas Renggli, Tudor Gîrba, and Oscar Nierstrasz. Embedding languages without breaking tools. In Theo D'Hondt, editor, *Proceedings of the 24th European Conference on Object-Oriented Programming (ECOOP'10)*, volume 6183 of *LNCS*, pages 380–404. Springer-Verlag, 2010.

[RHV+10] David Röthlisberger, Marcel Härry, Alex Villazón, Danilo Ansaloni, Walter Binder, Oscar Nierstrasz, and Philippe Moret. Exploiting dynamic information in ides improves speed and correctness of software maintenance tasks. *Transactions on Software Engineering*, 2010. To appear.

[RN09] Jorge Ressia and Oscar Nierstrasz. Dynamic synchronization — a synchronization model through behavioral reflection. In *Proceedings of International Workshop on Smalltalk Technologies (IWST 2009)*, pages 101–106, New York, NY, USA, 2009. ACM.

[RND09] David Röthlisberger, Oscar Nierstrasz, and Stéphane Ducasse. Autumn leaves: Curing the window plague in IDEs. In *Proceedings of the 16th Working Conference on Reverse Engineering (WCRE 2009)*, pages 237–246, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[RNDB09] David Röthlisberger, Oscar Nierstrasz, Stéphane Ducasse, and Alexandre Bergel. Tackling software navigation issues of the Smalltalk IDE. In *Proceedings of International Workshop on Smalltalk Technologies (IWST 2009)*, pages 58–67, New York, NY, USA, 2009. ACM.

[Röt09] David Röthlisberger. Why and how to substantiate the good of our reverse engineering tools? In *FAMOOSr, 3rd Workshop on FAMIX and Moose in Reengineering*, 2009.

[Röt10] David Röthlisberger. Exploiting dynamic information in ides eases software maintenance. In *Proceedings of the 5th International Workshop on Program Comprehension through Dynamic Analysis (PCODA 2010)*, pages 20–24. Technische Universiteit Delft, 2010.

[RRGN10] Jorge Ressia, Lukas Renggli, Tudor Gîrba, and Oscar Nierstrasz. Run-time evolution through explicit meta-objects. In *Proceedings of the 5th Workshop on Models@run.time at the ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems (MODELS 2010)*, October 2010.

[SWK10] Niko Schwarz, Erwann Wernli, and Adrian Kuhn. Hot clones, maintaining a link between software clones across repositories. In *IWSC '10: Proceedings of the 4th International Workshop on Software Clones*, pages 81–82, New York, NY, USA, April 2010. ACM.

[VBG+10] Toon Verwaest, Camillo Bruni, David Gurtner, Adrian Lienhard, and Oscar Nierstrasz. Pinocchio: Bringing reflection to life with first-class interpreters. In *OOPSLA Onward! '10*, 2010.

Theses and Student projects

[Fri10] Manuel Friedli. ECCrawler — visualizations for Eclipse. Informatikprojekt, University of Bern, October 2010.

[Gys10] Florian S. Gysin. Trust this Code? — improving code search results through human trustability factors. Bachelor's thesis, University of Bern, March 2010.

[Hae10] Marcel Haerry. Augmenting eclipse with dynamic information. Master's thesis, University of Bern, May 2010.

[Ren10] Lukas Renggli. *Dynamic Language Embedding With Homogeneous Tool Support*. Phd thesis, University of Bern, October 2010.

[Röt10] David Röthlisberger. *Augmenting IDEs with Runtime Information for Software Maintenance*. PhD thesis, University of Bern, May 2010.