Final Scientific Report
SNF Project no. 200020-113342
"Analyzing, capturing and taming software change"

October 31, 2008

a) Summary of results

In this project we investigated means to understand and enable the evolution of software systems. Some of the most significant results include:

1. Changeboxes: We have reported on a prototype implementation of a system for managing changes to running software applications, and we have explored ways to dynamically adapt the behaviour of software entities in response to changes in the run-time context.

2. Scoped Reflection: We have refined our approach to partial behavioural reflection to exploit a runtime model of the source code. This enables very fine-grained reflection at the sub-method level, which is useful for application ranging from detailed dynamic analysis to language extensions such as pluggable type systems.

3. Object Flow Analysis: We have applied our infrastructure for object flow analysis to aid in the post-hoc development of unit tests for existing applications, and we have extended our support for object flow analysis to the virtual machine level, thus making novel applications such as back-in-time debuggers practical and efficient.

4. Evolution Analysis: We have explored various ways of bringing information about the run-time behaviour of software entities and the long-term evolution of software to the developer, and we have begun to assess how well this information can support common developer tasks.

Results

The following results were obtained during the period from 2007-10-01 to 2008-09-30.

Changeboxes

Changeboxes offer the means to track and manage changes in a running software system. Depending on the context of the end-user, different versions of the same software entities may be visible to different users of the same deployed software system. We have reported on a prototype implementation of Changeboxes which effectively demonstrates the feasibility of the approach for scenarios where development of a running software system can proceed without danger of disturbing current clients [DGL+07].

We have explored extensions of the core ideas of Changeboxes to *context-oriented programming* (COP). COP provides programming language and run-time mechanisms to support the development of context-aware applications [HCN08]. Such applications can dynamically adapt their behaviour to the current run-time context, including properties of the application itself, or its run-time environment, or even external attributes, such as time-of-day, service availability, and so on. Changeboxes can be seen as a form of COP,

where the context in question is the current changebox, and the adaptation is the set of changes visible from that changebox.

We have explored mechanisms to support COP in dynamic languages such as Python [vLDN07]. We have also started to explore the formal semantics of COP by developing a calculus of evolving objects in which objects may adapt their behaviour at run-time [DCGN08].

Scoped Reflection

We refined our approach to support unanticipated partial behavior using bytecode manipulation [RDT08], and we used this infrastructure (i) to implement a novel approach to controlling the visibility of traits (fine-grained components for composing classes) [DWBN07], and (ii) to realize a novel approach for introducing software transactional memory to an existing programming language [RN07].

We came, however, to recognize that bytecode is at too low a level of abstraction to properly support reflection at the sub-method level. As a consequence, we have introduced a novel approach to represent the system's code not as text, but as an abstract syntax tree which is causally linked to its compiled form. This enabled full support for sub-method reflection as we could link reflective behavior to any node in the abstract syntax tree [Den08].

We have used this mechanism for several applications. We showed how we can scale the identification of features by dynamically analyzing parts of the system. Furthermore, the dynamic nature of our system allowed us to plug and unplug instrumentation at runtime [DGN07].

We also developed an approach to providing pluggable types in dynamic languages [HDN09]. To achieve this, we annotated parts of the code with type information that in turn were taken into account by various type systems for reasoning about different properties of software.

One well-known problem with reflective systems is how to handle infinite meta-call recursion that arises when reflecting on code used by core language features. (For example, if instrumentation code naively uses itself code that is instrumented, an endless loop will result.) We extended our sub-method reflection framework with the notion of *context* to allow us to distinguish between the regular level and meta level, and thus to prevent infinite meta-call recursion [DSD08].

Object Flow Analysis

Object Flow Analysis complements classical dynamic analysis of execution traces by representing object aliases as first class entities. We have specified our meta-model and described the analysis space that our approach opens [LDG07].

We have used object aliases to identify side effects in execution traces [LGGN07]. We have also used this information to guide the development of unit tests [LGGN08]. We use side effect information to first reverse engineer how tests should be set up, and afterwards to suggest possible assertions.

The most impressive result consisted in the use of object aliases to provide a practical solution for back-in-time debugging [LGN08]. Back-in-time debugging keeps all relevant information from the execution and makes it available for debugging. However, the quantity of information generated in implementations by other researchers made the approach impractical for real usage. Our solution was to introduce the concept of object aliasing at the virtual machine level, and to let the garbage collector delete no longer relevant information, thus keeping, in most cases, the memory usage within a practical limit. This work received a distinguished paper award at ECOOP 2008.

Evolution Analysis

In this work we explore how information about the evolution of software and information about the dynamic behaviour of software entities at run-time can be analyzed and fed back to the developer through the IDE. We have developed an experimental IDE, called *Hermion*, which feeds runtime information back to the developer, and we have carried some some initial user experiments to assess the usefulness of this information for typical developer tasks [RGN08] [Röt08b].

Hermion was awarded a European Smalltalk User Group Innovation Technology Award [Röt08a].

We have also collaborated with colleagues at the Swinburne University of Technology in Melbourne, Australia who analyze the evolution of multiple versions of large software projects at the bytecode level. This work has somewhat surprisingly revealed that the distribution of common software metrics stays relatively stable over time, even as a software system grows. Furthermore, most software entities do not change over time, but the most complex entities are the ones that are most likely to be modified [VSN07] [VSNW08].

A technical report summarizes our current thinking on the impact of these research threads, and has served as the basis for our followup SNF project ("Bringing Models Closer to Code", SNF project #200020-121594) [NDG+08]. (A revised and extended version of this technical report is scheduled for publication in the context of the new project.)

Staff contributions

All of the researchers funded by this project have made impressive progress over the past year.

- Adrian Kuhn has completed the second year of his PhD. He developed an approach to make explicit the dependencies between unit tests by means of example objects [KRH+08], and he supervised a student who implemented a framework, called JExample, that realizes this approach [Hae08]. He supervised another student who extended Java with features to manipulate sets of objects collectively [Ern08].

  He also organized and participated in a collective effort to implement a Smalltalk virtual machine using the PyPy infrastructure [BKL+08]. His current PhD research is focussing on ways to develop new kinds of programming abstractions from very fine-grained components. This work is closely aligned with the followup SNF project.

- Adrian Lienhard is in his last year of his PhD. He continued his work on Object Flow Analysis and applied it in several contexts. He has provided a detailed description of the approach showing how it complements classical approaches to dynamic analyses [LDG07]. He used it to expose the side effects in execution traces [LGGN07] and he used this information to guide the development of unit tests [LGGN08]. He also used the concept of object flow to modify the virtual machine to provide a practical approach for back-in-time debugging [LGN08]. This work received the distinguished paper award at ECOOP 2008, one of the most prestigious conferences in this domain. He is defending his PhD in December 2008.

- Lukas Renggli has completed the second year of his PhD. He participated in supervising the work on Changeboxes [DGL+07]. He also started to work on a model for transactional memory that can also be used as a mechanism for managing the evolution of the state of objects [RN07]. This work has led to his current PhD topic, which is concerned with embedding domain models in code, and is described in more detail in the followup SNF project mentioned above.

- David Röthlisberger has completed the second year of his PhD. He is investigating the integration of both existing and novel analysis techniques into the development environment. He has reported on the role of querying dynamic information from the IDE [Röt08b], and he has showed how dynamic analyses can be integrated in the IDE [RGN08]. He also participated in the work on handling unanticipated partial behavioral reflection through bytecode manipulation [RDT08].

  He supports his experiments by demonstrating them in a novel IDE called Hermion [Röt08a], which received a European Smalltalk User Group Innovation Technology Award at the 2008 International Smalltalk Conference[Röt08a].

Changes to the research plan

No major changes have occurred in the research plan.

Important events

- Oscar Nierstrasz presented an invited talk on "Model-Centric Software Adaptation", which summarized many of the results of this SNF project, at SVPP 2008 (Software Variability: a Programmers' Perspective – Brussels, Belgium, August 8-9, 2008).
- Lukas Renggli was Invited Speaker at ISC 2008 (International Smalltalk Conference – Amsterdam, August 25-29)
- Adrian Kuhn was Invited Speaker at ISC 2008 (International Smalltalk Conference – Amsterdam, August 25-29)
- Marcus Denker presented the paper "The Meta in Meta-object Architectures" at TOOLS 2008 (Zurich, June 30 - July 4, 2008)
- Lukas Renggli presented "Transactional Memory for Smalltalk" at ICDL 2007 (International Conference on Dynamic Languages)
- Adrian Lienhard presented
  - "Test Blueprints – Exposing Side Effects in Execution Traces to Support Writing Unit Tests" at CSMR 2008 (European Conference on Maintenance and Reengineering)
  - "Practical Object-Oriented Back-in-Time Debugging" at ECOOP 2008 (European Conference on Object-Orienter Programming)
- Adrian Kuhn presented "Back to the future in one week — implementing a Smalltalk VM in PyPy" at the Workshop on Self-sustaining Systems (S3) 2008
- David Röthlisberger presented
  - "Exploiting Runtime Information in the IDE" at ICPC 2008 (International Conference on Program Comprehension)
  - "Querying Runtime Information in the IDE" at QTAPC 2008 (Working Session on Query Technologies and Applications for Program Comprehension co-located with ICPC 2008)
  - "Hermion - Exploiting the Dynamics of Software" at ESUG Innovation Technology Awards 2008
- Oscar Nierstrasz was PC Member of
  - Models@run.time (Colocated with Models 2008 – Toulouse, France, Sept 28-Oct 3, 2008)
  - PC Member of DLS 2008 (Dynamic Languages Symposium – Paphos, Cyprus, July 8, 2008)
  - PC Member of WASDeTT 2008 (International Workshop on Advanced Software Development Tools and Techniques – Paphos, Cyprus, July 8, 2008)
  - PC Member of TOOLS Europe 2008 (International Conference on Objects, Models, Components, Patterns – Zurich, Switzerland, June 30 - July 4, 2008)
  - PC Member of ICPC 2008 (International Conference on Program Comprehension – Amsterdam, The Netherlands, 10-13 June, 2008)
  - PC Member of FASE 2008 (Fundamental Approaches to Software Engineering – Budapest, Hungary, 29 March - 6 April, 2008)
  - DLS 07 (Dynamic Languages Symposium 2007 — colocated with OOPSLA 07, Montreal, Oct. 22, 2007)
- Oscar Nierstrasz *co-organized* the CHOOSE Forum 2007 (Languages for the Web – FHNW Brugg, Nov. 9, 2007)

- Marcus Denker *co-organized* ISC 2008 (International Smalltalk Conference – Amsterdam, Aug. 25-29, 2008)
- Adrian Kuhn *co-organized* International Workshop on Advanced Software Development Tools and Techniques (WASDeTT) co-located with ECOOP 2008.
- David Röthlisberger *co-organized* PCODA 2008 (4th Workshop on Program comprehension through dynamic analysis) co-located with WCRE 2008.
- Oscar Nierstrasz served in Editorial Boards:
  - Springer LNCS – SL2 – Programming Techniques and Software Engineering (Series Editor)
  - ACM TOSEM – Transactions on Software Engineering and Methodology (Associate Editor)

b) Publications

Published papers are annexed to this report. They are all available electronically as PDF files at the following url:

`http://www.iam.unibe.ch/~scg/cgi-bin/scgbib.cgi?snf08`

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

Published papers

[DCGN08] Mariangiola Dezani-Ciancaglini, Paola Giannini, and Oscar Nierstrasz. A calculus of evolving objects. In *Proceedings of the 6th International Workshop on Multiparadigm Programming with Object-Oriented Languages (MPOOL 2008)*, 2008.

[DGL+07] Marcus Denker, Tudor Gîrba, Adrian Lienhard, Oscar Nierstrasz, Lukas Renggli, and Pascal Zumkehr. Encapsulating and exploiting change with Changeboxes. In *Proceedings of the 2007 International Conference on Dynamic Languages (ICDL 2007)*, pages 25–49. ACM Digital Library, 2007.

[DGN07] Marcus Denker, Orla Greevy, and Oscar Nierstrasz. Supporting feature analysis with runtime annotations. In *Proceedings of the 3rd International Workshop on Program Comprehension through Dynamic Analysis (PCODA 2007)*, pages 29–33. Technische Universiteit Delft, 2007.

[DSD08] 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, 2008.

[DWBN07] Stéphane Ducasse, Roel Wuyts, Alexandre Bergel, and Oscar 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)*, pages 171–190, New York, NY, USA, October 2007. ACM Press.

[HCN08] Robert Hirschfeld, Pascal Costanza, and Oscar Nierstrasz. Context-oriented programming. *Journal of Object Technology*, 7(3), March 2008.

[HDN09] Niklaus Haldimann, Marcus Denker, and Oscar Nierstrasz. Practical, pluggable types for a dynamic language. *Journal of Computer Languages, Systems and Structures*, 35(1):48–64, April 2009.

[KRH+08] Adrian Kuhn, Bart Van Rompaey, Lea Haensenberger, Oscar Nierstrasz, Serge Demeyer, Markus Gaelli, and Koenraad Van Leemput. JExample: Exploiting dependencies between tests to improve defect localization. In P. Abrahamsson, editor, *Extreme Programming and Agile Processes in Software Engineering, 9th International Conference, XP 2008*, Lecture Notes in Computer Science, pages 73–82. Springer, 2008.

[LDG07] Adrian Lienhard, Stéphane Ducasse, and Tudor Gîrba. Object flow analysis — taking an object-centric view on dynamic analysis. In *Proceedings of the 2007 International Conference on Dynamic Languages (ICDL'07)*, pages 121–140, New York, NY, USA, 2007. ACM Digital Library.

[LGGN07] Adrian Lienhard, Tudor Gîrba, Orla Greevy, and Oscar Nierstrasz. Exposing side effects in execution traces. In Andy Zaidman, Abdelwahab Hamou-Lhadj, and Orla Greevy, editors, *Proceedings of the 3rd International Workshop on Program Comprehension through Dynamic Analysis (PCODA'07)*, pages 11–17. Technische Universiteit Delft, 2007.

[LGGN08] Adrian Lienhard, Tudor Gîrba, Orla Greevy, and Oscar Nierstrasz. Test blueprints – exposing side effects in execution traces to support writing unit tests. In *Proceedings of the 12th European Conference on Software Maintenance and Reengineering (CSMR'08)*, pages 83–92. IEEE Computer Society Press, 2008.

[LGN08] Adrian Lienhard, Tudor Gîrba, and Oscar Nierstrasz. Practical object-oriented back-in-time debugging. In *Proceedings of the 22nd European Conference on Object-Oriented Programming (ECOOP'08)*, volume 5142 of *LNCS*, pages 592–615. Springer, 2008. ECOOP distinguished paper award.

[NDG+08] Oscar Nierstrasz, Marcus Denker, Tudor Gîrba, Adrian Kuhn, Adrian Lienhard, and David Röthlisberger. Self-aware, evolving eternal systems. Technical Report IAM-08-001, University of Bern, Institute of Applied Mathematics and Computer Sciences, 2008.

[RDT08] 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.

[RGN08] David Röthlisberger, Orla Greevy, and Oscar Nierstrasz. Exploiting runtime information in the ide. In *Proceedings of the 16th International Conference on Program Comprehension (ICPC 2008)*, volume 0, pages 63–72, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[RN07] Lukas Renggli and Oscar Nierstrasz. Transactional memory for Smalltalk. In *Proceedings of the 2007 International Conference on Dynamic Languages (ICDL 2007)*, pages 207–221. ACM Digital Library, 2007.

[Röt08a] David Röthlisberger. Hermion — exploiting the dynamics of software. European Smalltalk User Group Innovation Technology Award, August 2008.

[Röt08b] David Röthlisberger. Querying runtime information in the IDE. In *Proceedings of the 2008 workshop on Query Technologies and Applications for Program Comprehension (QTAPC 2008)*, page n4, 2008.

[vLDN07] Martin von Löwis, Marcus Denker, and Oscar Nierstrasz. Context-oriented programming: Beyond layers. In *Proceedings of the 2007 International Conference on Dynamic Languages (ICDL 2007)*, pages 143–156. ACM Digital Library, 2007.

[VSN07] Rajesh Vasa, Jean-Guy Schneider, and Oscar Nierstrasz. The inevitable stability of software change. In *Proceedings of 23rd IEEE International Conference on Software Maintenance (ICSM '07)*, pages 4–13, Los Alamitos CA, 2007. IEEE Computer Society.

[VSNW08] Rajesh Vasa, Jean-Guy Schneider, Oscar Nierstrasz, and Clint Woodward. On the resilience of classes to change. In Tom Mens, Maja D'Hondt, and Kim Mens, editors, *Proceedings of 3d International ERCIM Symposium on Software Evolution (Software Evolution 2007)*, volume 8. Electronic Communications of the EASST, 2008.

Theses and Student projects

[Den08] Marcus Denker. *Sub-method Structural and Behavioral Reflection*. Phd thesis, University of Bern, May 2008.

[Ern08] David Erni. JAG — a prototype for collective behavior in Java. Bachelor's thesis, University of Bern, August 2008.

[Hae08] Lea Haensenberger. JExample. Bachelor's project, University of Bern, March 2008.

c) Publications in press

Publications to appear

[BKL+08] Carl Friedrich Bolz, Adrian Kuhn, Adrian Lienhard, Nicholas D. Matsakis, Oscar Nierstrasz, Lukas Renggli, Armin Rigo, and Toon Verwaest. Back to the future in one week — implementing a Smalltalk VM in PyPy. In *Workshop on Self-sustaining Systems (S3) 2008*, 2008. To appear.