Intermediate Scientific Report
SNF Project no. 2000-067855.02
"Tools and Techniques for Decomposing and Composing Software"

November 10, 2003

a) Summary of results

This project addresses the problem of how to organize and structure software systems in such a way that they can be easily adapted to changing requirements. We focus on (1) tools and techniques for extracting architectural artifacts, *i.e.*, for *decomposing* software, and (2) mechanisms and language features for flexibly constructing software from parts, *i.e.*, for *composing* software. The key results in the first year include (1) techniques for extracting behavioural dependencies in legacy software using Concept Analysis, and for visualizing and understanding run-time structures, and (2) the development of innovative programming language features for building object-oriented software from fine-grained units of reuse (*traits*), for specifying extensions to existing software bases in a local context (*Classboxes*), and specifying applications as compositions of components (*Piccola*).

This project is carried out under close collaboration with *RECAST: Evolution of Object-Oriented Applications* (SNF Project No. 620-066077). Whereas RECAST concentrates more generally on modeling of object-oriented software, program understanding and software evolution, this project focuses on technical issues related to object-oriented languages and language design.

Results

The results obtained in the first year correspond closely to the project workplan, with a few additions. First we present results pertaining to program understanding (decomposition), and then we consider results dealing with language design (composition).

Software Decomposition

Considerable work has been done to extend and refine the MOOSE reengineering platform. MOOSE is a general-purpose tool for loading, modeling and analyzing models of software systems. Software systems implemented in different languages can be parsed with a variety of different tools, and loaded into MOOSE. These models can then stored, queried, analyzed and visualized. MOOSE itself is not a research result, but it is a key component of our laboratory, since it enables the rapid development of reverse engineering and reengineering tools needed to carry our research experiments. As such, there are no publications on MOOSE itself, but rather on the experiments that MOOSE enables.

One very successful research track has been the application of Formal Concept Analysis (FCA), a technique for detecting recurring patterns (*i.e.*, "concepts") in sets of elements and their properties, to models of software. The MOOSE platform has been used here to carry out experiments in which FCA was applied to detect recurring behavioural patterns in object-oriented class hierarchies [Aré03a] [Buc03]. Further experiments are in progress, and further publications are in press [ADN03b] [ADN03a]. FCA has also been applied to the understanding of the inner workings of classes [Aré03b]. This work has been previously reported in the context of the RECAST project (No. 620-066077).

A static analysis of software source code will not necessary reveal the run-time architectural artifacts. We have applied and extended the techniques of lightweight visualization developed in Lanza's PhD thesis [Lan03] to the visualization of run-time structures [BDL03]. Further publications are also pending. This work is described in greater detail in a Master's thesis carried out in collaboration with the I3S Laboratory of the University of Nice at Sophia-Antipolis [Ber03].

Visualization and querying of software models have also been combined to identify meaningful groups of software artifacts [Tal03]. The results presented in this master's thesis are now integrated in CodeCrawler, the software visualisation platform that is based on top of MOOSE [Lan03].

As planned, the logic programming framework SOUL has been integrated into MOOSE. This will enable more advanced forms of querying for MOOSE models, which is needed for further experiments related to architectural extraction and validation of architectural constraints [Aeb03]. SOUL has been used to represent and codify software architecture. SOUL acts as a foundation for logic meta programming, where logic programming is used to reasoning on software [MWVM03]. We are currently using SOUL to express advanced behavioral tests over the trace of programs and further publications will follow.

Software Composition

We have also achieved several results in the area of constructing flexible software systems from components. This work includes both refinements of older, more mature research tracks (Piccola, OpenCoLaS), and new, innovative directors (Traits, Classboxes).

Piccola is an experimental language for composing applications from software components, and JPiccola is the Java-based implementation of the language. The language definition and its semantics are stable and mature, and a publication is pending [NA03a]. A JPiccola manual and tutorial are now available, and are included in the web download[1] [NAK03]. Newer results include the development of an experimental type system for Piccola [Nie03], the definition of compositional styles based on this type system [Kne03], and a technique for converting white box, inheritance-based software reuse to black-box, component-based reuse by means of run-time creation of classes [Sch03].

OpenCoLaS [Cru02] is a framework for evaluating coordination models, a complementary approach to composing components in a distributed environment.

We have also started a series of experiments in new directions.

Traits offer a fine-grained model of software reuse that allow duplicated code to be cleanly refactored without the need for complex models of multiple inheritance [SDNB03]. A first formalization of the traits model has been elaborated [SND+02], and a detailed description has been submitted for publication. First experiments applying traits to refactoring a non-trivial class hierarchy have been carried out [BSD02] and a full paper will appear shortly [BSD03]. A paper on the interactive environment (the traits browser) is in the publication pipeline [SB03].

[1] www.iam.unibe.ch/∼scg/Research/Piccola

Classboxes offer a coarse-grained model of software reuse offering local class extensions [BDW03b] [BDW03a]. A Classbox is a simple module system in which sets of classes may be imported from or exported to other Classboxes. Classes may be locally extended without impacting other Classboxes. Classboxes offer an elegant way to uniformly extend a set of cooperating classes – a problem that cannot be solved by subclassing. A formalization of the Classbox model is underway.

Surfaces are a further exploration of the theme of feature visibility in object-oriented languages [DSW03].

OOPAL is a model that unifies array programming and object-oriented language features [DM03] [MD03]. F-script, the language that implements the OOPAL model, is freely available and is attracting increasing exposure as it is integrated with the Mac OS X operating system.

Staff contributions

- Gabiela Arévalo is carrying out the experiments with Formal Concept Analysis. She has developed a tool, ConAn, which applies FCA algorithms to MOOSE models [Aré03a] [Aré03b] [ADN03b].
- Alexandre Bergel has developed the Classbox model and its implementation [BDW03b] [BDW03a].
- Laura Ponisio is working on a technique to modularize software based on existing dependencies between artifacts. She has developed a tool, Baobab, which analyzes coupling and cohesion within MOOSE models. The detected dependencies will be used to improve the clustering of artifacts within modules. This work is relatively new, and no publication are available yet.
- Nathanael Schaerli has developed the traits model and its implementation [BSD02] [BSD03] [SND+02] [SDNB03] [SB03].

Changes to the research plan

No major changes have occurred in the research plan.

Important events

- The PhD thesis of Michele Lanza [Lan03] has been awarded the prestigious Denert-Stiftung Software Engineering prize for 2003. (See: `www.denert-stiftung.de`)
- Oscar Nierstrasz was an Invited Speaker at FMCO 2002 (First International Symposium on Formal Methods for Components and Objects –Leiden, The Netherlands, Nov. 5-8, 2002)
- We have presented papers in a series of high-profile conferences and workshops: ECOOP 2003 (European Conference on Object-Oriented Programming), JMLC 2003 (Joint Modular Languages Conference), Coordination 2003, LMO 2003 (Langages et Modèles à Objets).
- We have organized two workshops at ECOOP 2003: *the 4th International Workshop on Object-Oriented Reengineering*, and *Object-Oriented Language Engineering for the Post-Java Era*
- We have organized a workshop on declarative Meta Programming (Edinburgh, 2003) [MWVM03].

b) Publications

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

www.iam.unibe.ch/∼scg/cgi-bin/oobib.cgi?snf03

References

[Aré03a] Gabriela Arévalo. Understanding behavioral dependencies in class hierarchies using concept analysis. In *Proceedings of LMO 2003: Langages et Modeles à Objets*, pages 47–59. Hermes, Paris, January 2003.

[BDL03] Roland Bertuli, Stéphane Ducasse, and Michele Lanza. Run-time information visualization for understanding object-oriented systems. In *Proceedings of WOOR 2003 (4th International Workshop on Object-Oriented Reengineering)*, pages 10–19. University of Antwerp, 2003.

[BDW03a] Alexandre Bergel, Stéphane Ducasse, and Roel Wuyts. The classbox module system. In *Proceedings of the ECOOP '03 Workshop on Object-oriented Language Engineering for the Post-Java Era*, July 2003.

[BDW03b] Alexandre Bergel, Stéphane Ducasse, and Roel Wuyts. Classboxes: A minimal module model supporting local rebinding. In *Proceedings of the Joint Modular Languages Conference 2003*, volume 2789 of *LNCS*, pages 122–131. Springer-Verlag, 2003.

[BSD02] Andrew Black, Nathanael Schärli, and Stéphane Ducasse. Applying traits to the Smalltalk collection hierarchy. Technical Report IAM-02-007, Institut für Informatik, Universität Bern, Switzerland, November 2002. Also available as Technical Report CSE-02-014, OGI School of Science & Engineering, Beaverton, Oregon, USA.

[Cru02] Juan-Carlos Cruz. OpenCoLaS – a coordination framework for CoLaS dialects. In *Proceedings of COORDINATION 2002*, York, United Kingdom, 2002.

[DM03] Stéphane Ducasse and Philippe Mougin. Power to collections: Generalizing polymorphism by unifying array programming and object-oriented programming. In *Proceedings of the ECOOP '03 Workshop on Object-oriented Language Engineering for the Post-Java Era*, July 2003.

[DSW03] Stéphane Ducasse, Nathanael Schaerli, and Roel Wuyts. Open surfaces for controlled visibility. In *Proceedings of the ECOOP '03 Workshop on Object-oriented Language Engineering for the Post-Java Era*, July 2003.

[MWVM03] Tom Mens, Roel Wuyts, Kris De Volder, and Kim Mens. Workshop proceedings — declarative meta programming to support software development. *ACM SIGSOFT Software Engineering Notes*, 28(1), January 2003.

[NAK03] Oscar Nierstrasz, Franz Achermann, and Stefan Kneubuehl. A guide to JPiccola. Technical Report IAM-03-003, Institut für Informatik, Universität Bern, Switzerland, June 2003.

[Nie03] Oscar Nierstrasz. Contractual types. Technical Report IAM-03-004, Institut für Informatik, Universität Bern, Switzerland, 2003.

[SDNB03] Nathanael Schärli, Stéphane Ducasse, Oscar Nierstrasz, and Andrew Black. Traits: Composable units of behavior. In *Proceedings ECOOP 2003*, LNCS, pages 248–274. Springer Verlag, July 2003.

[SND+02] Nathanael Schärli, Oscar Nierstrasz, Stéphane Ducasse, Roel Wuyts, and Andrew Black. Traits: The formal model. Technical Report IAM-02-006, Institut für Informatik, Universität Bern, Switzerland, November 2002. Also available as Technical Report CSE-02-013, OGI School of Science & Engineering, Beaverton, Oregon, USA.

Theses

The following Masters' and PhD theses are *not* included with this report, but are available at the following url:

www.iam.unibe.ch/∼scg/cgi-bin/oobib.cgi?snf03

References

[Aeb03] Tobias Aebi. Extracting architectural information using different levels of collaboration. Diploma thesis, University of Bern, September 2003.

[Ber03] Roland Bertuli. Compréhension de systemes orientés-objet par l'utilisation d'informations dynamiques condensées. Master's thesis, École Supérieure en Science Informatiques, Sophia-Antipolis, France, 2003.

[Buc03] Frank Buchli. Detecting software patterns using formal concept analysis. Diploma thesis, University of Bern, September 2003.

[Kne03] Stefan Kneubuehl. Typeful compositional styles. Diploma thesis, University of Bern, April 2003.

[Lan03] Michele Lanza. *Object-Oriented Reverse Engineering — Coarse-grained, Fine-grained, and Evolutionary Software Visualization*. PhD thesis, University of Berne, May 2003.

[Sch03] Andreas Schlapbach. Enabling white-box reuse in a pure composition language. Diploma thesis, University of Bern, January 2003.

[Tal03] Daniele Talerico. Grouping in object-oriented reverse engineering. Diploma thesis, University of Bern, June 2003.

RECAST publications

The following papers have been published in the context of the RECAST project, and are *not* included with this report. They have been previously submitted with the intermediate report for RECAST. Electronic versions are available at:

www.iam.unibe.ch/∼scg/cgi-bin/oobib.cgi?recast03

References

[Aré03b] Gabriela Arévalo. X-Ray views on a class using concept analysis. In *Proceedings of WOOR 2003 (4th International Workshop on Object-Oriented Reengineering)*, pages 76–80. University of Antwerp, July 2003.

[Lan03b] Michele Lanza. Codecrawler — lessons learned in building a software visualization tool. In *Proceedings of CSMR 2003*, pages 409–418. IEEE Press, 2003.

c) Publications in press

The following papers have been accepted for publication, and will be included in the final report for this project.

www.iam.unibe.ch/∼scg/cgi-bin/oobib.cgi?snf04

References

[ADN03a] Gabriela Arévalo, Stéphane Ducasse, and Oscar Nierstrasz. Understanding classes using X-Ray views. In *Proceedings of 2nd International Workshop on MASPEGHI 2003 (ASE 2003)*, pages 9–18. CRIM - University of Montreal (Canada), October 2003.

[ADN03b] Gabriela Arévalo, Stéphane Ducasse, and Oscar Nierstrasz. X-Ray views: Understanding the internals of classes. In *Proceedings of ASE 2003*, pages 267–270. IEEE Computer Society, October 2003.

[BSD03] Andrew P. Black, Nathanael Schärli, and Stéphane Ducasse. Applying traits to the Smalltalk collection hierarchy. In *Proceedings OOPSLA 2003*, October 2003. To appear.

[MD03] Philippe Mougin and Stéphane Ducasse. OOPAL: Integrating array programming in object-oriented programming. In *Proceedings OOPSLA 2003*, October 2003.

[NA03a] Oscar Nierstrasz and Franz Achermann. A calculus for modeling software components. In *FMCO 2002 Proceedings*, LNCS. Springer-Verlag, 2003. To appear.

[NA03b] Oscar Nierstrasz and Franz Achermann. Separating concerns with first-class namespaces. In Tzilla Elrad, Siobán Clarke, Mehmet Aksit, and Robert Filman, editors, *Aspect-Oriented Software Development*. Addison-Wesley, 2003. To appear.

[SB03] Nathanael Schärli and Andrew P. Black. A browser for incremental programming. *Computer Languages, Systems and Structures*, 2003. (To appear, special issue on Smalltalk).