Final Scientific Report — NFS Project no. 20-61655.00
"Meta-models and Tools for Evolution Towards Component Systems"

November 5, 2002

a) Summary of results

This project was concerned with developing tools and models to support the transition towards component-based software development. Results achieved in this project can be grouped according to the themes of the original project proposal:

- Towards a Component Meta Model
- Component Migration
- Compositional Infrastructure

Towards a Component Meta Model

First we consider results related to *modeling*, *manipulating* and *reasoning* about software systems in order to better understand them, *i.e.*, in order to support reverse engineering.

In order to understand and evolve a software system, appropriate *models* are needed. We have developed a meta-model for characterizing software entities called FAMIX and a software "repository" based on this meta-model, called MOOSE. This repository is effectively a platform for exploring software models, and forms the basis of several of the experimental tools developed in the course of this project. [DL01] describes a general methodology for program understanding based on this meta-model and platform.

One of the key issues in program understanding is to extract useful information from large amounts of software data. Software metrics can be extremely effective in processing this data, and visualization of metrics helps one to quickly get an overview of the data. CodeCrawler is a generic metrics visualization tool developed within the scope of this project, and applied to large software systems. [ML02] describes a generic graph-based model on top of which metrics can be extracted and [LD02a] presents the metrics that have been applied in reverse engineering experiments in the context of CodeCrawler.

Metrics-based visualization works best when applied to simple, direct metrics. In order to recognize and reason about relationships between software entities, however, other approaches are needed. SOUL is a declarative meta-programming language which has been integrated with MOOSE. SOUL has been reimplemented to improve the efficiency, and a library written in SOUL to do static reasoning of Smalltalk programs was refactored. SOUL is now used by around 20 people in 4 countries that use Declarative Meta Programming as the engine to drive research in software engineering regarding metrics, aspect-oriented programming, framework documentation, evolution, and program understanding. [MMW01b] [MMW01a] present an overview of the work done. [WD01] describes how SOUL is used to reason about object-oriented software artifacts.

In order to understand a software application, the developer must know how different software artifacts are related. Very often, however, these relationships are implicit rather than explicit. We started to apply *concept analysis*, a branch of lattice theory, which allows us to group elements based on common properties, to discover such implicit relationships. In [AM02a, AM02b] we apply concept analysis to well known Smalltalk class hierarchies. Specifically we studied how the classes in an object-oriented inheritance hierarchy are coupled by means of the inheritance and interface relationships. In this work we identify implicit *concept patterns* that express how reuse is achieved, identify weak spots in the hierarchy that could be improved, and recognize guidelines for customizing and extending the hierarchy.

We have also investigated ways of using dynamic trace information as an additional source of reverse engineering information. In [RD02, Ric02] we describe how such dynamic information can be used effectively to uncover roles and collaborations during architectural recovery.

Finally, in [Sch02] we have explored how different navigational aids and mechanisms can help a developer during the reverse engineering process.

Component Migration

The second track in this project explored how systems evolve.

[DDN02] is a book which presents guidelines and techniques for reverse and reengineering as a collection of so-called *reengineering patterns*. Most of the material for this book was gathered during earlier projects, particularly SNF-2000-46947.96 and BBW-96.0015, which are acknowledged in the preface to the book, though the process of writing and editing was completed during the current project.

[Duc01] is a habilitation which describes in further detail the tools and techniques used in reengineering, and which summarizes much of the work carries out in this and earlier, related SNF projects.

Newer work includes the following:

- [Wuy01] describes how SOUL can be used to reason about changes in an evolving software system.
- [LD02b] explores the use of CodeCrawler to visualize metrics that compare different versions of a software system.
- [Ste01] explores the use of a query engine to recover refactoring which have been performed on a software system.
- [KN01] investigates a series of refactorings which can be used to eliminate duplicated code.
- [Tic01] explores the broader issue of modeling and tool support for language-independent refactoring.
- Finally, [Kau01] explores the sources of problems arises during the evolution of large software systems, and [Nie02] proposes a research agenda that focuses on software evolution.

Compositional Infrastructure

The final track of the project concerns infrastructure for building software systems from components.

Most of this work has focused on *Piccola*, an experimental composition language, which has been extensively explored in previous projects. [Ach02] describes *Piccola* in detail, providing a formal semantics for the language, and describing how this semantics can be used to aid reasoning about properties of software compositions. We have also explored a variety of other languages and approaches for software composition.

[Hof01] explores the use of a coordination medium to dynamically reconfigure heterogeneous compositions of software components.

[SLN01] surveys the suitability of scripting and coordination languages for coordinating cooperating software agents.

In this track we also explore new ways of structuring code and a new object model. First we have been working on the definition and integration of dynamic interfaces as a way to convey role intention which is important for composition [SD02]. Then, more fundamentally, we have been defining a new object model that introduces the notion of *traits* (groups of methods) [SDN02]. Traits allow one to compose classes out of reusable abstractions. Traits are orthogonal to inheritance which is the traditional way of reusing object-oriented code. We envision traits as the basic brick on top of which we will build a more advanced composition and component language.

PECOS: a Companion Research Project

Dr. Roel Wuyts joined the lab in December 2000. The bulk of his research focussed on developing a component model for embedded devices in the context of the IST project PECOS [GC02]. This project was concerned with bringing component-based software engineering to a particular kind of small embedded systems. The component model copes with the different constraints imposed by the environment (low power consumption, little memory, simple CPUs) [NA02] and supports the verification of non-functional requirements (checking of timing and scheduling issues) [WD02], [LA02].

SNF Personnel

The following research personnel were funded fully or in part by the SNF project:

- Franz Achermann developed the semantics and implementation of *Piccola* [Ach02].
- Stéphane Ducasse was the architect of the MOOSE environment, and supervised various projects that built on MOOSE [DDN02], [DL01], [Duc01], [RD02]. Ducasse also co-supervised the work of Arévalo [AM02a] [AM02b], Golomingi [KN01], Hofmann [Hof01], Richner [RD02], Schweizer [Sch02], and Steiger [Ste01].
- Michele Lanza developed CodeCrawler [LD02b] [LD02a] [ML02].
- Sander Tichelaar was largely responsible for the FAMIX meta-model that MOOSE is based on, and he developed the language-independent refactoring support of MOOSE [Tic01].
- Roel Wuyts developed SOUL and integrated it into MOOSE [MMW01b] [MMW01a] [WD01] [Wuy01].

b) Publications

Copies of the following publications accompany the final report.

References

[AM02a] Gabriela Arévalo and Tom Mens. Analysing object oriented application frameworks using concept analysis. In Andrew Black, Erik Ernst, Peter Grogono, and Markky Sakkinen, editors, *ECOOP 2002: Proceedings of the Inheritance Workshop*. University of Jyväskylä, 2002.

[AM02b] Gabriela Arévalo and Tom Mens. Analysing object oriented framework reuse using concept analysis. In Jean-Michel Bruel and Zohra Bellahsene, editors, *Advances in Object-oriented information systems: OOIS 2002 Workshops*. Springer Verlag, 2002.

[DDN02] Serge Demeyer, Stéphane Ducasse, and Oscar Nierstrasz. *Object-Oriented Reengineering Patterns*. Morgan Kaufmann, 2002.

[DL01] Stéphane Ducasse and Michele Lanza. Towards a methodology for the understanding of object-oriented systems. *Technique et science informatiques*, 20(4):539–566, 2001.

[LD02a] Michele Lanza and Stéphane Ducasse. Beyond language independent object-oriented metrics: Model independent metrics. In Fernando Brito e Abreu, Mario Piattini, Geert Poels, and Houari A. Sahraoui, editors, *Proceedings of the 6th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, pages 77–84, 2002.

[LD02b] Michele Lanza and Stéphane Ducasse. Understanding software evolution using a combination of software visualization and software metrics. In *Proceedings of LMO 2002*, pages 135–149, 2002.

[ML02] Tom Mens and Michele Lanza. A graph-based metamodel for object-oriented software metrics. *Electronic Notes in Theoretical Computer Science*, 72(2), 2002.

[MMW01a] K. Mens, I. Michiels, and R. Wuyts. Supporting software development through declaratively codified programming patterns. *SEKE 2001 Special Issue of Elsevier Journal on Expert Systems with Applications*, 2001. Extended version of [MMW01b].

[MMW01b] Kim Mens, Isabel Michiels, and Roel Wuyts. Supporting software development through declaratively codified programming patterns. In *SEKE 2001 Proceedings*, pages 236–243. Knowledge Systems Institute, 2001. International conference on Software Engineering and Knowledge Engineering, Buenos Aires, Argentina, June 13-15, 2001.

[Nie02] Oscar Nierstrasz. Software evolution as the key to productivity. In *Proceedings Radical Innovations of Software and Systems Engineering in the Future*, Venice, Italy, Oct. 2002. to appear.

[RD02] Tamar Richner and Stéphane Ducasse. Using dynamic information for the iterative recovery of collaborations and roles. *Proceedings of ICSM'2002 (International Conference on Software Maintenance)*, October 2002.

[SD02] Benny Sadeh and Stéphane Ducasse, Adding Dynamic Interface to Smalltalk, Journal of Object Technology, vol. 1, no. 1, 2002.

[SDN02] Nathanael Schaerli, Stéphane Ducasse and Oscar Nierstrasz, Classes = Traits + States + Glue (Beyond mixins and multiple inheritance), Proceedings of the International Workshop on Inheritance, 2002.

[SLN01] Jean-Guy Schneider, Markus Lumpe, and Oscar Nierstrasz. Agent coordination via scripting languages. In Andrea Omicini, Franco Zambonelli, Matthias Klusch, and Robert Tolksdorf, editors, *Coordination of Internet Agents*, pages 153–175. Springer-Verlag, 2001.

[WD01] Roel Wuyts and Stéphane Ducasse. Symbiotic reflection between an object-oriented and a logic programming language. In *ECOOP 2001 International workshop on MultiParadigm Programming with Object-Oriented Languages*, 2001.

[Wuy01] Roel Wuyts. Synchronising changes to design and implementation using a declarative meta-programming language. In *International Workshop on (Constraint) Logic Programming for Software Engineering*, Dec. 2001.

Theses, Habilitation

The following theses and habilitation are not included with the report, but are all available from the following url:

`http://www.iam.unibe.ch/~scg/cgi-bin/oobib.cgi?snf02`

References

[Ach02] Franz Achermann. *Forms, Agents and Channels - Defining Composition Abstraction with Style*. PhD thesis, University of Berne, January 2002.

[Duc01] Stéphane Ducasse. Reengineering object-oriented applications. Technical report, Université Pierre et Marie Curie (Paris 6), 2001. Habilitation à diriger des recherches.

[Hof01] Thomas F. Hofmann. OPENSPACES, an object-oriented framework for configurable coordination of heterogeneous agents. Diploma thesis, University of Bern, April 2001.

[Kau01] Christian Kaufmann. Software engineering im spannungsfeld theorie und praxis. Master's thesis, University of Bern, 2001.

[KN01] Georges Golomingi Koni-N'sapu. A scenario based approach for refactoring duplicated code in object oriented systems. Diploma thesis, University of Bern, June 2001.

[Ric02] Tamar Richner. *Recovering Behavioral Design Views: a Query-Based Approach*. PhD thesis, University of Berne, May 2002.

[Sch02] Daniel Schweizer. Navigation in object-oriented reverse engineering. Diploma thesis, University of Bern, June 2002.

[Ste01] Lukas Steiger. Recovering the evolution of object oriented software systems using a flexible query engine. Diploma thesis, University of Bern, June 2001.

[Tic01] Sander Tichelaar. *Modeling Object-Oriented Software for Reverse Engineering and Refactoring*. PhD thesis, University of Berne, December 2001.

PECOS papers

The following publications are related to the PECOS project. Since this project was funded from a different source (BBW), we do not include copies of these papers. Most are available, however, from the following url:

`http://www.iam.unibe.ch/~scg/cgi-bin/oobib.cgi?pecos`

References

[GC02] Thomas Genssler, Alexander Christoph, Benedikt Schulz, Michael Winter, Chris M. Stich, Christian Zeidler, Peter Müller, Andreas Stelter, Oscar Nierstrasz, Stéphane Ducasse, Gabriela Arévalo, Roel Wuyts, Peng Liang, Bastiaan Schönhage, and Reinier van den Born , PECOS in a Nutshell, The PECOS Consortium, September 2002.

[LA02] Peng Liang, Gabriela Arévalo, Stéphane Ducasse, Michele Lanza, Nathanael Schaerli, Roel Wuyts and Oscar Nierstrasz, Applying RMA for Scheduling Field Device Components, ECOOP 2002 Workshop Reader, 2002.

[NA02] Oscar Nierstrasz, Gabriela Arévalo, Stéphane Ducasse, Roel Wuyts, Andrew Black, Peter Müller, Christian Zeidler, Thomas Genssler, and Reinier van den Born, A Component Model for Field Devices, Proceedings First International IFIP/ACM Working Conference on Component Deployment, ACM, Berlin, Germany, June 2002, pp. 200-209.

[WD02] Roel Wuyts and Stéphane Ducasse, Non-Functional Requirements in a Component Model for Embedded Systems, International Workshop on Specification and Verification of Component-Based Systems, 2001

c) Publications in press

None