RECAST: Evolution of Object-Oriented Applications

SNF 620-066077

Intermediate Scientifique Report

September 2003 - August 2004

15 Octobre 2004

The goal of the Recast project is to support the evolution of object-oriented applications by focusing on three main directions: reverse engineering and reengineering, versions analysis, and migration towards components. As we are part of the Software Composition Group of the University of Bern, we worked on topics related to the Recast project but also on topics related to the project "Tools and Techniques for Decomposing and Composing Software"[1](TTDCS in the rest of this document). Therefore this report also lists the results obtained in the context of language design. Note that they are listed here to indicate that we obtained other results but they will be described in detail in the TTDCS scientific report.

1 Recast Results

1.1 Publications

The results of the Recast project for the current period can be categorized into three large areas: a reengineering environment development, program visualizaton and application analysis.

Reengineering Environment. We continued the development of MOOSE, our reengineering environment [9] [14]. In particular now we are able to extend it easily to support new analyses such as dynamic information [6] and evolution analysis [16]. We plan to refine and unify its meta-model to support code generation. Our environment is now distributed with the CD of the VisualWorks distribution of Smalltalk developed by Cincom, http://www.cincom.com/smalltalk). To the best of our knowledge our environment is used by one company in Germany, Smalltalked-Visuals GmbH and consultantsof other companies. MOOSE is also used by the following research groups: LORE (Prof. Demeyer University of Antwerp), DECOMP (Prof. Wuyts, Université Libre de Bruxelles) and following researchers: Dr. Lanza (University of Lugano), Prof. K. Mens (University of Louvain-la-neuve).

[1](SNF Project No. 2000-067855.02, Oct. 2002 - Sept. 2004)

We worked on the identification of the key infrastructural design variations that have to be taken into account when developing reengineering environments. Indeed, most of the time reengineering environments are built without a clear analysis of the impact of the representation choices. Our analysis serves as a basis to understand the concerns that have to be taken into account when designing new reengineering environments [2].

Program Visualizations. We continued our work on the definition of new visualizations to support program understanding [1][4][6][8].

Polymetric Views. Reverse engineering has become a major concern in software industry because of the sheer size and complexity of software systems. This problem needs to be tackled, since the systems in question are of considerable worth to their owners and maintainers. We developed present the concept of a *polymetric view*, a lightweight software visualization technique enriched with software metrics information [1] [6]. Polymetric views help to understand the structure and detect problems of a software system in the initial phases of a reverse engineering process. We discuss the benefits and limits of several predefined polymetric views we have implemented in our tool CodeCrawler. Moreover, based on clusters of different polymetric views we have developed a methodology which supports and guides a software engineer in the first phases of a reverse engineering of a large software system. We have refined this methodology by repeatedly applying it on industrial systems, and illustrate it by applying a selection of polymetric views to a case study. Note that Software-Tomography a professional environment started to use this idea.

Code Duplication Understanding. We defined new visualizations to support the understanding of duplicated code analysis reports [12]. Duplication of code is a common phenomenon in the development and maintenance of large software systems. The detection and removal of duplicated code has become a standard activity during the refactoring phases of a software life-cycle. However, code duplication identification tends to produce large amounts of data making the understanding of the duplication situation as a whole difficult. Reengineers can easily lose sight of the forest for the trees. There is a need to support a qualitative analysis of the duplicated code. We proposed a number of visualizations of duplicated source elements that support reengineers in answering questions, *e.g.,* which parts of the system are connected by copied code or which parts of the system are copied the most. This work will be described in the dissertation of Mr. M. Rieger and the submission of a journal paper.

Run-Time Behavior. Understanding the run-time behavior of object-oriented legacy systems is a complex task due to factors such as late binding and polymorphism [4]. Current approaches extract and use information from the *complete* execution trace of a system. The sheer size and complexity of such traces make their handling, storage, and analysis difficult. Current software systems which run almost non-stop do not permit such a full analysis. We have developed a lightweight approach based on the extraction of a condensed amount of information, *e.g.,*

measurements, that does not require a full trace. Using this condensed information, we propose a visualization approach which allows us to identify and understand certain aspects of the objects' lifetime such as their role played in the creation of other objects and the communication architecture they support.

Analysis

Recurrent Structural Coding Idioms. We continued the work based on formal concept analysis to support reverse engineering [11]. The idea is to use concept analysis to identify recurrent structural or behavior patterns in object-oriented applications. This work resulted in the master thesis of Mr. F. Buchli and is extended in the dissertation of Mrs G. Arevalo.

Design Flaw Identification. As systems evolve and their structure decays, maintainers need accurate and automatic identification of the design problems. Current approaches for automatic detection of design problems are not accurate enough because they analyze only a single version of a system and consequently they miss essential information as design problems appear and evolve over time. We have developed an approach that uses the historical information of the suspected flawed structures to increase the accuracy of the automatic problem detection. Our means is to define measurements which summarize how persistent the problem was and how much maintenance effort was spent on the suspected structure [5].

Trace-based and Logic-Based Testing. We experimented with the use of logic to express tests in object-oriented applications [7]. We propose to represent the trace of object-oriented applications as logic facts and express tests over the trace. This way complex sequences of message exchanges, sequence matching, or expression of negative information are expressed in compact form. This work resulted in the master thesis of Mr. M. Friedig.

Evolution Analysis. We started to work on the analysis of trends in evolution [8] [10] [15]. Knowing where to start reverse engineering a large software system, when no information other than the system's source code itself is available, is a daunting task. Having the history of the code (i.e., the versions) could be of help if this would not imply analyzing a huge amount of data.We developed an approach for identifying candidate classes for reverse engineering and reengineering efforts. Our solution is based on summarizing the changes in the evolution of object-oriented software systems by defining history measurements. This analysis is based on the retrospective empirical observation that classes which changed the most in the recent past also suffer important changes in the near future.

We started to work on the analysis of packages and their remodularisation and will have soon some results.

1.2 Contributions of Collaborators

Mr. Gîrba refined the HisMo model to analyze evolution trends in large object-oriented systems. We are now ready to perform experiments to validate our

hypotheses. We started to have papers accepted in major conferences such as International conference on Software Maintenance, and we are preparing publications for international journal.

We exchanged Mr. Gälli and Mrs Ponisio with the TTDC project because there was a better match between the projects and their research topics. Mrs Ponisio is working on assessing packages and supporting modularisation of object-oriented applications. We are preparing some publications.

1.3 Network and Project Participation

In addition to our participation in the RELEASE ESF network on Software Evolution and the EVOL Belgium funded network of research, we are currently participating in a new ERCIM working group on Software Evolution.

The project, Traits whose goal is to evaluate the possibility to introduce traits in C# has been accepted in the context of the Microsoft ROTOR Shared Source program. A Microsoft source mentions that the acceptance rate was 16% but such an information will not be publicly available because of Microsoft Research strategy.

1.4 Organized Events

- Organization of two workshops at the European Conference on Object-Oriented Programming ECOOP'2004: Object - Oriented Language Engineering in Post - Java Era and Object - Oriented Reengineering.
- Organization of the Annual European Smalltalk User Group Conference (110 participants). Chair of the Academic Track, editor of the special issue of the journal Computer Languages, Systems and Structures from Elsevier.
- Participation in the RELEASE ESF network, the Foundation of Software Evolution network and the ERCIM working group on Software Evolution.

Workshop Proceedings.

Sebastian Gonzales, Wolgang Demeuter, Pascal Costanza, Stéphane Ducasse, Richard Gabriel and Theo D'hondt, Report of the ECOOP'04 Workshop on Object-Oriented Language Engineering in Post-Java Era, 2004. LNCS (Lecture Notes in Computer Science), Springer - Verlag, 2004.

Roel Wuyts, Serge Demeyer, Stéphane Ducasse and Kim Mens, Report of the ECOOP'04 Workshop on Object - Oriented Reengineering, *In Object - Oriented Technology (ECOOP'04 Workshop Reader)*, LNCS (Lecture Notes in Computer Science), Springer - Verlag, 2004.

2 Publications

As we are part of the Software Composition Group of the University of Berne, we worked on topics that are related to the evolution of object-oriented applications but

also on topics related to the project "Tools and Techniques for Decomposing and Composing Software" (SNF Project No. 2000-067855.02, Oct. 2002 - Sept. 2004) (TTDCS).

To avoid discriminating one or other of the projects, we decided to clearly separate the publications. We selected as RECAST publications results related to reengineering and evolution publications and we selected as TTDCS results related to language design to TTDCS. For these papers we only list them here and they will be joined to the TTDCS report.

2.1 Recast Publications

Note that some of the publications below were listed in the inprint section of the report of last period since they were not presented or published at the deadline report date.

[1] Michele Lanza and Stéphane Ducasse, Polymetric Views – A Lightweight Visual Approach to Reverse Engineering, *IEEE Transactions on Software Engineering*, vol. 29, no. 9, September 2003, pp. 782–795.

[2] Stéphane Ducasse and Sander Tichelaar, Dimensions of Reengineering Environment Infrastructures, *International Journal on Software Maintenance: Research and Practice*, pp. 345-373, Vol 15, Oct, 2003.

[3] Roel Wuyts and Stéphane Ducasse, Unanticipated Integration of Development Tools using the Classification Model, Journal of Computer Languages, Systems and Structures, vol. 30, no. 1-2, 2004, pp. 6377.

[4] Stéphane Ducasse, Michele Lanza and Roland Bertuli, High-Level Polymetric Views of Condensed Run-Time Information, *Proceedings of CSMR 2004 (Conference on Software Maintenance and Reengineering)*, 2004, pp. 309 - 318.

[5] Daniel Raţiu, Stéphane Ducasse, Tudor Gîrba and Radu Marinescu, Using History Information to Improve Design Flaws Detection, *Proceedings of CSMR 2004 (European Conference on Software Maintenance and Reengineering)*, 2004, pp. 223–232.

[6] Michele Lanza, Program Visualization Support for Highly Iterative Development Environments, *Proceedings of VisSoft 2003 (International Workshop on Visualizing Software for Understanding and Analysis)*, IEEE CS Press, 2003, pp. 62 – 67.

[7] Stéphane Ducasse, Michael Freidig and Roel Wuyts, Logic and Trace-based Object-Oriented Application Testing, *Fifth International Workshop on Object-Oriented Reengineering (WOOR 2004)*, 2004.

[8] Tudor Gîrba and Michele Lanza, Visualizing and Characterizing the Evolution of Class Hierarchies, *Fifth International Workshop on Object-Oriented Reengineering (WOOR 2004)*, 2004.

[9] Oscar Nierstrasz and Stéphane Ducasse, Moose a Language-Independent Reengineering Environment, ERCIM News, vol. 58, July 2004, pp. 24-25.

In Print. We got articles accepted that are in print or that have been published after the 1st of September 2004. They will be joined in the 2004-2005 report.

[10] Tudor Gîrba, Stéphane Ducasse and Michele Lanza, Yesterday's Weather: Guiding Early Reverse Engineering Efforts by Summarizing the Evolution of Changes, *Proceedings of ICSM 2004 (International Conference on Software Maintenance)*, 2004.

[11] Gabriela Arévalo, Frank Buchli and Oscar Nierstrasz, Detecting Implicit Collaboration Patterns, *Proceedings of WCRE 2004 (11th Working Conference on Reverse Engineering)*, IEEE Computer Society Press, November 2004, to appear.

[12] Matthias Rieger, Stéphane Ducasse and Michele Lanza, Insights into System-Wide Code Duplication, *Proceedings of WCRE 2004 (11th Working Conference on Reverse Engineering)*, IEEE Computer Society Press, November 2004, to appear.

[13] Stéphane Ducasse, Tudor Gîrba, Michele Lanza and Serge Demeyer, Moose: a Collaborative and Extensible Reengineering Environment, *Reengineering Environments*, 2004, Chapter in book to appear.

[14] Michele Lanza and Stéphane Ducasse, CodeCrawler An Extensible and Language Independent 2D and 3D Software Visualization Tool, *Reengineering Environments*, 2004, Chapter in book to appear.

[15] Tudor Gîrba, Stéphane Ducasse, Radu Marinescu and Daniel Raţiu, Identifying Entities That Change Together, *Ninth IEEE Workshop on Empirical Studies of Software Maintenance*, 2004, to appear.

[16] Stéphane Ducasse, Tudor Gîrba and Jean-Marie Favre, Modeling Software Evolution by Treating History as a First Class Entity, *Workshop on Software Evolution Through Transformation (SETra 2004)*, 2004, to appear.

Projects and Masters

[17] Thomas Bühler, MooseGager, a Software Metrics Tool based on Moose, Technical Report, University of Bern, October 2003, Informatikprojekt.

[18] Michael Freidig, Trace Based Object-Oriented Application Testing, Masters thesis, University of Bern, January 2004.

[19] Christoph Wysseier, CCJun Polymetric Views in Three-dimensional Space, Technical Report, University of Berne, June 2004, Informatikprojekt.

2.2 Other Publications

Here are the publications that we obtain in the context of the TTDCS project. They are mainly related to new language features. Again we distinguish between printed and in print papers.

[20] Philippe Mougin and Stéphane Ducasse, OOPAL: Integrating Array Programming in Object - Oriented Programming, *In OOPSLA'2003 (International Conference on Object - Oriented Programming Systems Languages and Applications)*, 2003.

[21] Andrew P. Black, Nathanael Schärli and Stéphane Ducasse, Applying Traits to the Smalltalk Collection Hierarchy, *Proceedings OOPSLA 2003*, 2003.

[22] Roel Wuyts, Stéphane Ducasse and Oscar Nierstrasz, A Data-centric Approach to Composing Embedded, Real-time Software Components, To appear in *Elsevier Journal of Systems and Software — Special Issue on Automated Component-Based Software Engineering*, 2003.

[23] Nathanael Schärli, Stéphane Ducasse, Oscar Nierstrasz and Roel Wuyts, Composable Encapsulation Policies, Proceedings ECOOP 2004 (European Conference on Object-Oriented Programming), LNCS 3086, Springer Verlag, June 2004, pp. 248274.

In print

[24] Alexandre Bergel, Christophe Dony and Stéphane Ducasse, Prototalk: an Environment for Teaching, Understanding, Designing and Prototyping Object-Oriented Languages, Proceedings of ESUG 2004, September 2004, To appear.

[25] Alexandre Bergel and Stéphane Ducasse, Dynamically Scoped Aspects with Classboxes, Proceedings of the First JFDPA (Journée franaise de la programmation par aspects), September 2004, To appear in *L'Objet* a french-speaking journal.

[26] Alexandre Bergel, Stéphane Ducasse, Oscar Nierstrasz and Roel Wuyts, Classboxes: Controlling Visibility of Class Extensions, To appear in *Elsevier Journal of Computer Languages, Systems and Structures*.

[27] Stéphane Ducasse, Nathanael Schärli and Roel Wuyts, Uniform and Safe Metaclass Composition, To appear in *Elsevier Journal of Computer Languages, Systems and Structures*.