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” (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 visualization 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 consultants of 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).
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. Gödba 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.


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

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.


**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.


**Projects and Masters**


### 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.


**In print**


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