Final Scientific Report
Hasler Project no. 2234
"Enabling the evolution of J2EE applications through reverse engineering and quality assurance'

October 30, 2010

a) Summary of results

Enterprise applications are typically complex and thus require elaborate techniques to support their understanding. This project focuses on developing dedicated reverse engineering techniques for such applications. The most significant results regard the meta-modeling of enterprise applications and the interactive presentation of the information in these applications.

Results

We present the results obtained during the final period from 2009-08-01 to 2010-10-30. For a summary of earlier results, please consult the intermediate reports available from the project web site[1].

Case Studies

Enterprise systems are typically proprietary. Thus, to study them we need access to case studies provided by industrial partners. During the past year we continued to work closely with the Eidgenössisches Institut für Geistiges Eigentum (IGE). They provided us with numerous real-life problems that they frequently encounter, and we developed several new analyses that represent solutions to their problems. The issues of concern typically lie in the area of quality assurance, with problems spanning from identifying unsafe database queries to identifying inadequate layering.

Meta-modeling

Java Enterprise Applications are complex software systems that contain heterogeneous data and are implemented as a blend of different technologies. To analyze these systems we first need to create a meta-model to represent their structure and the relations among the different components. In the past years we have developed a meta-model that covers Java Enterprise Beans, the structure of databases and the mapping between the code and the database. We are now in the process of developing other extensions to the meta-model in order to include into it also elements related to the UI (*e.g.*, Java Server Pages) and elements related to the build system (*e.g.*, Maven).

Once the addition of these extensions will be complete, we will be able to represent entirely a Java Enterprise Application and analyze the interplay between the various components deployed together. The development of the meta-model did not start from scratch, but instead we extended FAMIX, a language-independent meta-model for source code analysis.

[1] `http://scg.unibe.ch/research/hasler07`

Parsing

In parallel with the meta-modelling work we are building fact extractors for the various sources: Java Server Pages, Maven configuration files and SQL files. To support the fast implementation of different parsers for each different programming languages, members of our team have implemented PetitParser, a parser framework which combines ideas from scannerless parsing, parser combinators, parsing expression grammars and packrat parsers to model grammars and parsers as objects that can be reconfigured dynamically [RDGN10]. PetitParser was originally developed in the context of our SNF project, "Bringing Models Closer to Code" (SNF Project No. 200020-121594, Oct. 2008 - Sept. 2010)[2]. We applied the PetitParser framework for defining several parsers for languages used in the enterprise ecosystems and by doing this, demonstrated that PetitParser is not only flexible but also practical.

Visualizations

To understand enterprise systems we need to visualize the data in ways that expose the complexity of their structure. One of the critical problems raised by the heterogeneous nature of Java Enterprise Applications (JEAs) is identifying transaction scope. We developed visualizations which clearly expose which methods of a JEA are involved in a transaction (*i.e.*, transaction flow visualization), and which methods that access the database through JDBC are not involved in any transaction (*i.e.*, unsafe query visualization) [PGN10].

Two key architectural constraints are given by the distribution of the remote services and the persistency of data. In the Java Enterprise Edition platform (JEE), this is typically supported through SessionBeans and EntityBeans respectively. In many systems, although the base language is nominally object-oriented, the design of the resulting system is nevertheless procedural: the behaviour typically lies in the SessionBeans, while the data reside in the EntityBeans and DTOs. These systems would benefit from a refactoring towards a stronger business model that unifies behavior and data and to which SessionBeans would delegate responsibilities. Our strategy is to expose the DTOs and the way they are used by the different services. In order to achieve this task we developed a visualization called *DTOs Constellation* that helps in the initial phase of identifying such a rich object model [PG10].

Figure 1 shows the visualization as applied to an industrial content management system (CMS) to manage customer data. Gray squares represent DTOs while white squares are Java classes that invoke some methods of the DTOs. A light gray edge represents an invocation from one class to another while a black edge represents code duplication. The graph is then laid out using a force-based layout that reveals clusters of connected entities.

The visualization is drawn using the Mondrian visualization engine, while the code duplication is retrieved by SmallDude, a duplication detector included in the Moose analysis platform. In our example, the DTOs are visually clustered in 3 groups each composed of several classes. Manual inspection of the code verifies that these elements actually have a common context and all together collaborate to manage logically related data. Interestingly, the classes in the middle of Figure 1 are connecting two different clusters of elements. These classes are actually Session Beans that need to collect data from two different contexts exposing a relation between them that would not be evident otherwise.

The most interesting exposed elements are the code duplications expressed by the black edges. There are three kinds of code duplication present: (1) duplication between classes using the same DTO, (2) duplication between DTOs, and (3) duplications between classes of two different clusters. The first kind of duplication suggests the presence of code that could be factored out from the classes suffering from duplication. The second kind of duplication can highlight the presence of DTOs sharing the same data or sharing the same logic to manage different data. A further inspection of these elements can drive the merging of two DTOs into one, or the redistribution of the management of the data and the data itself.

By investigating non-evident relations among software elements and using this information to drive the refactoring, the third kind of code duplication can highlight the presence of the same business logic being used to manage different kind of data. The duplications between classes using the DTOs relate to some shared business logic that could be factored out. In contrast, the duplication between two DTOs reveals shared data that can be factored out in a smarter DTO hierarchy.

[2] `http://scg.unibe.ch/research/snf08`

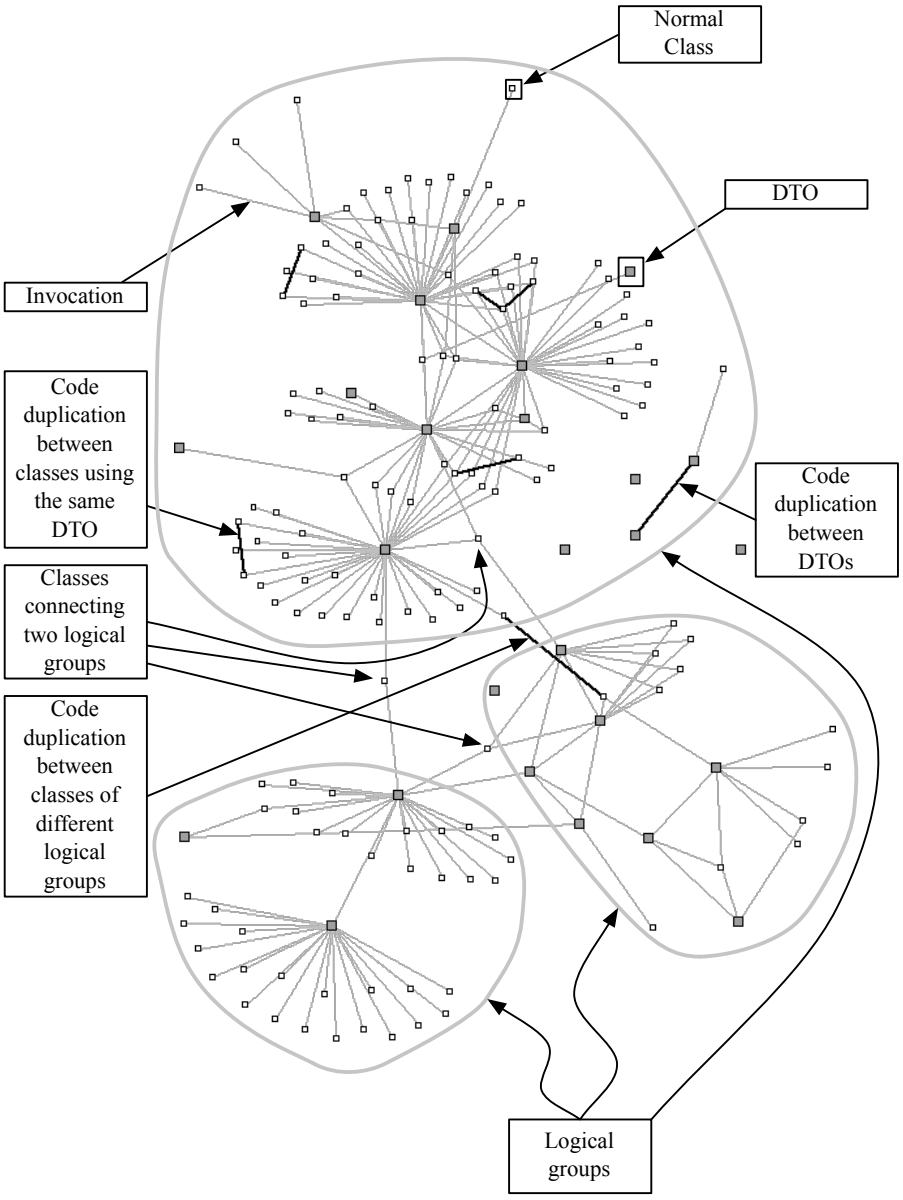

Figure 1: DTOs Constellation visualization with highlighting on the main concepts

Interactive Browsers and Software Understanding

While visualizations are useful tools for revealing the structure of unknown data, the complexity of enterprise applications requires us to explore the various facets using interactive browsers. Based on Glamour, a browser scripting framework, we started the development of an Enterprise Application Browser. Figure 2 presents the browser while exploring one of our case studies. The browser is organized as follows: the first panel contains the list of the packages in a project, the second contains the list of classes contained in the selected package, the third contains the list of methods of the selected class and the *class blueprint* visualization. In the panel below it is possible to see the source code of the selected class or method and the transaction flow of the hierarchy containing the selected class. In the first three panels the elements that are part of a transaction scope or part of an unsafe path [PGN10] that access the database are annotated with a gray label.

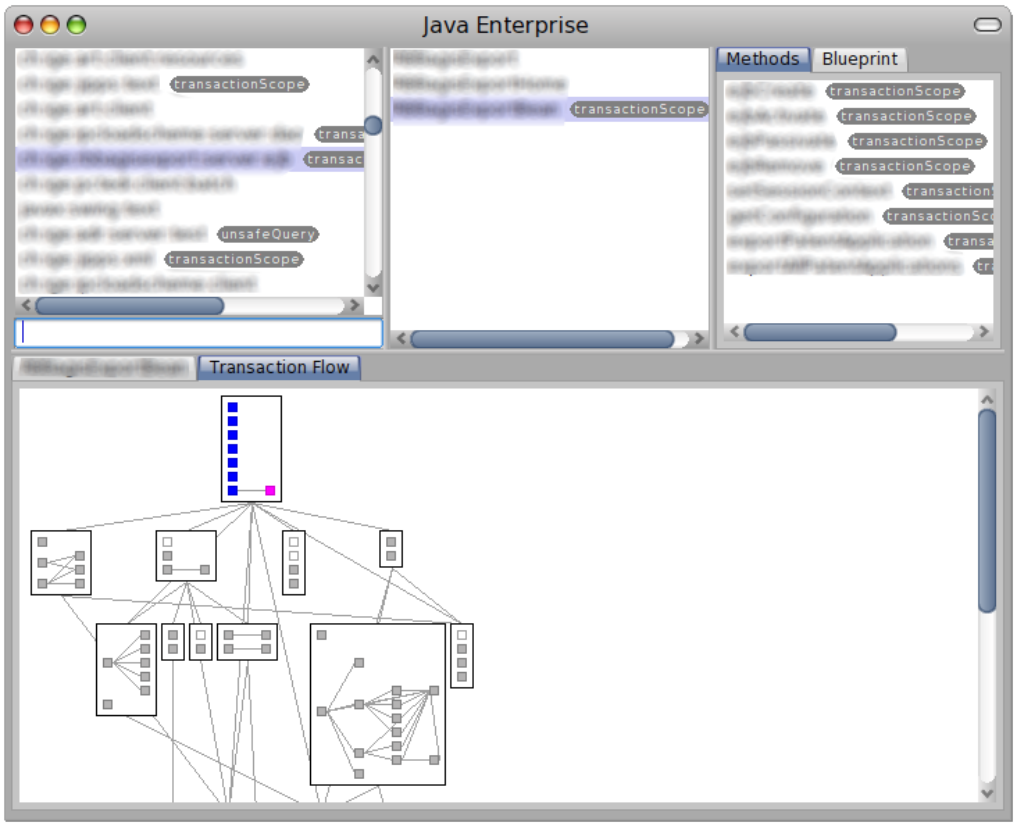

Figure 2: Code browser for Java Enterprise applications (package, class and method names obscured due to NDA)

Using annotations it is possible to immediately identify all the elements of interest in the model. Moreover, having an instrument that merges software visualizations with the code, helps the overall comprehension of the enterprise application.

We plan to extend this implementation with other visualizations and annotations on other aspects of the code for example with information regarding the deployment of the application.

One of the drawbacks of this approach is that it can not be automated: the reverse engineer needs to analyze the resulting visualization. To find a solution to this, we also explored one way of automatically suggesting starting points for the analysis. Our approach is inspired by PageRank and it aims at assigning weights to code entities relative to their importance in the system. PageRank was originally developed by Larry Page and Sergei Brin to rank search results of the Google internet search engine. We implemented this approach in Smalltalk, and we called it Code Rank [PRR10]. We applied Code Rank on a large number of case-studies in Smalltalk and Java proving that approach can be effective in supporting the reverse engineer in the identification of the most important parts of the code. Our approach can be applied to various kinds of software artifacts and relations at different levels of abstraction.

Staff contributions

- Fabrizio Perin started his third year of the PhD. He refined the meta-model for analyzing Java Enterprise Applications and developed several dedicated analyses and visualizations. He worked on the refinement of the already investigated aspect of application transaction scope. He developed an extension of Moose enabling the analysis of enterprise applications on the Moose platform. He worked together with Lukas Renggli and Jorge Ressia in the application of a well-known algorithm

used in the context of web search engine to investigate the relations among software artifacts. He actively worked with our industrial partner, Eidgenössisches Institut für Geistiges Eigentum (IGE), to identify design quality problems and develop corresponding solutions.

- Tudor Gîrba served as PostDoc. He left this position in October 2010, but he continued to participate as an external collaborator through to the end of the project.

Changes to the research plan

No major changes intervened in the research plan over the past year.

Important events

- Fabrizio Perin was:
  - PC Member and Session Chair of FAMOOSr 2010 (4th Workshop on FAMIX and Moose in Reengineering, Colocated with ICSM 2010, Timisoara, Romania, Sep. 12-18, 2010).
- Tudor Gîrba was:
  - Tool Demo Chair of ICSM 2010 (International Conference on Software Maintenance)
  - PC member of WCRE 2010 (Working Conference on Reverse Engineering).
  - PC member of MODELS 2010 (International Conference on Model Driven Engineering Languages and Systems).
  - PC member of MSR 2010 (Working Conference on Mining Software Repositories).
  - PC member of ENASE 2010 (International Conference on Evaluation of Novel Approaches to Software Engineering).
- Oscar Nierstrasz was:
  - Program Chair of CASTA 2009 (Workshop on Context-Aware Software Technology and Applications)
  - Co-organizer of FOSD 20010 (2nd Workshop on Feature-Oriented Software Development)
  - Invited Speaker at SOFSEM 2010 (36th International Conference on Current Trends in Theory and Practice of Computer Science, Principles of Software Construction track)
  - PC Member of FAMOOSr 2009 (3rd Workshop on FAMIX and Moose in Reengineering)
  - PC Member of FOSD 2009 (1st International Workshop on Feature-Oriented Software Development)
  - PC Member of IWST09 (International Workshop on Smalltalk Technologies)
  - PC Member of RAM-SE 2009 (Workshop on Reflection, AOP and Meta-Data for Software Evolution, colocated with ECOOP 2009)
  - PC Member of ECOOP 2009 (23rd European Conference on Object-Oriented Programming)
  - PC Member of ETSM 2009 (First International Symposium on Emerging Trends in Software Metrics)
  - PC Member of PCODA 2010 (Program Comprehension through Dynamic Analysis workshop)
  - PC Member of ICSM 2010 (26th IEEE International Conference on Software Maintenance)
  - PC Member of SEAA-EDISON 2010 (Euromicro Special track on Evolution of Distributed, Internet-based and Service-Oriented applicatioNs)
  - PC Member of TOOLS 2010 (48th International Conference on Objects, Models, Components, Patterns)

- PC Member of RAM-SE 2010 (7th ECOOP Workshop on Reflection, AOP and Meta-Data for Software Evolution)
- PC Member of Suite 2010 (2nd Intl. Workshop on Search-driven development: Users, Infrastructure, Tools and Evaluation)
- PC Member of WETSoM 2010 (Workshop on Emerging Trends in Software Metrics)
- PC Member of OOPS 2010 (Special track on Object-Oriented Programming Languages and Systems at SAC 2010)

b) Publications

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

`http://scg.unibe.ch/scgbib?query=hasler10`

Published papers

[DG09] Simon Denier and Tudor Gîrba. Workshop on FAMIX and Moose in software reengineering (FAMOOSr 2009). In *16th Working Conference on Software Maintenance and Reengineering (WCRE 2009)*, pages 325–326, October 2009.

[Per10] Fabrizio Perin. Moosejee: A moose extension to enable the assessment of jeas. In *Proceedings of the 26th International Conference on Software Maintenance (ICSM 2010) (Tool Demonstration)*, September 2010.

[PG10] Fabrizio Perin and Tudor Gîrba. Evaluating code duplication to identify rich business objects from data transfer objects. In *4th Workshop on FAMIX and Moose in Reengineering (FAMOOSr 2010)*, September 2010.

[PGN10] Fabrizio Perin, Tudor Gîrba, and Oscar Nierstrasz. Recovery and analysis of transaction scope from scattered information in Java enterprise applications. In *Proceedings of International Conference on Software Maintenance 2010*, September 2010.

[PRR10] Fabrizio Perin, Lukas Renggli, and Jorge Ressia. Ranking software artifacts. In *4th Workshop on FAMIX and Moose in Reengineering (FAMOOSr 2010)*, 2010.

[RDGN10] Lukas Renggli, Stéphane Ducasse, Tudor Gîrba, and Oscar Nierstrasz. Practical dynamic grammars for dynamic languages. In *4th Workshop on Dynamic Languages and Applications (DYLA 2010)*, Malaga, Spain, June 2010.