Introduction for the Fifth International Workshop on Object-Oriented Reengineering

Jonne Itkonen

Department of
Mathematical Information Technology
University of Jyväskylä Finland
E-mail: ji@mit.jyu.fi

We have built a course enrolment and study management system called Korppi for our own faculty, and little by little other faculties have started to use it also.

Korppi is a web-based system that has been in active use since it was released in autumn 2001. Ever since the system has been developed through several third year undergraduate projects each lasting for a semester. Each project has added more features to Korppi, introducing new working techniques and applying a coding style of their own.

Each developer or project group has worked on the specific modules of Korppi so the assumption is that the developers have edited only the modules they have been responsible for. However, they all have access to all parts of Korppi. As a result of the loose control of the distribution of the work and poor documentation of the development process, none of the developers has a clear picture of the system in detail. Furthermore, the structure of Korppi is believed to be corrupted and needs to be rediscovered. So, one can consider Korppi as a true, real world application, and not as an academic exercise.

Technically Korppi is a collection of Java Server Page files run on a Tomcat server. The data is saved in a PostgreSQL database. Most of the action in Java Server Page files is collected to JavaBeans. Usually there is one class to handle database connections and ideally another class collects all the SQL sentences needed in each subcomponent of Korppi.

We did a small analysis[1] to strength our believes about the erosion in Korppi. We used a method called *relation analysis* to find out hidden connections between classes in Korppi system. Basically, we found out which classes, or compilation units, have changed together (inside a small time period), and collected the number of same connections to a graph.

What we found out, was that, as expected, the system had few main modules, which didn't have strong couplings between them. We did find some god-classes, some duplicated responsibilities and one module inside a module. Of all of these we got glimpses from the analysis, but the code had still to be carefully manually examined. Relation analysis does not help in the examination itself, but it helps in choosing the areas of code that are to be examined.

Relation analysis is one of three phases in an analysing method called QCR, other two being *quantitative analysis* and *change sequence analysis*. We only did relation analysis, but the two other phases were partly covered by interviewing the developers of the system. It was nice to hear them suggesting that we should do something like quantitative analysis or change sequence analysis, even though we had not told them these analysis exist. We are now applying the whole QCR to Korppi, and also considering a large number of other possible metrics, available in literature, applied to Korppi to get more confidence with our presumptions. We believe, that the results of these metrics will also help us in finding the areas of Korppi, which require reengineering.

Metrics themselves are useless, if they are not interpreted correctly. I hope participating in WOOR will help me not only in how to reengineer object-oriented systems, but also to know when and why to reengineer them.

[1] This analysis is published in Proceedings of the Eight European Conference on Software Maintenance and Reengineering, pp. 233–239, IEEE Computer Society, 2004.