SQuAVisiT

Roubtsov, Serguei; Telea, Alexandru; Holten, Danny

Published in:
EPRINTS-BOOK-TITLE

SQuAVisiT: A Software Quality Assessment and Visualisation Toolset

Serguei Roubtsov, Alexandru Telea, Danny Holten
Department of Mathematics and Computer Science, Technische Universiteit Eindhoven
Den Dolech 2, 5600MB, Eindhoven, the Netherlands
s.roubtsov@tue.nl, a.c.telea@tue.nl, d.h.r.holten@tue.nl

Abstract

Software quality assessment of large COBOL industrial legacy systems, both for maintenance or migration purposes, mounts a serious challenge. We present the Software Quality Assessment and Visualisation Toolset (SQuAVisiT), which assists users in performing the above task. First, it allows a fully automatic extraction of metrics, call information, and code duplication from COBOL source code. This information, stored into a database, can be easily converted and exported to a set of visualization tools. We incorporated several such third-party tools for the visualization of call relations and system structure, and metrics visualization. These tools use novel visualization techniques such as bundled edges, matrix plots, and table lens. We illustrate the usage of our toolset with an industrial case study on a COBOL system comprising about 3000 modules and 1.7 million lines of code.

1. Introduction

Legacy COBOL systems usually contain millions of lines of code (LOC) spread over thousands of modules, developed by tens of people over many years, are often poorly documented and, to a large extent, knowledge about them is lost. Assessment of the code quality can help maintenance activities for such code. For example, a low quality code component is a prime, but also difficult, candidate for refactoring. A first step in quality assessment is typically the retrieval of structural and metric information, such as call graphs and source-level metrics. Although many tools such as parsers and fact extractors exist for this, this is still a semi-automatic, delicate process. Besides, interpretation of the retrieved facts, such as tens of thousands of metrics and module interdependencies, is also difficult. We combine automated software structure and quality information retrieval with visualisation of the obtained results in one toolset, called SQuAVisiT: Software Quality Assessment and Visualisation Toolset.

2. The Toolset

SQuAVisiT has two subsystems: one for quality assessment and the other for visualisation.

The quality assessment (QuA) subsystem is structured around a database which stores facts and metrics extracted by a number of third-party plug-ins. One such plug-in is a COBOL preprocessor and parser which also extracts basic facts, such as number of LOC, McCabe's cyclomatic complexity, and call relations. We implemented this plug-in using the `javacc` parser generator [3], which should help us targeting other languages besides COBOL. Another plug-in integrates CCFinder [4], a state of the art code duplication detection tool.

The visualization (Vis) subsystem couples the database with a number of visualisation tools, also loosely integrated as plug-ins. Four such tools are described below. TableVision [5] displays tables of hundreds of thousands of rows containing extraction metrics. MatrixZoom [1] displays dependency (*e.g.* call graphs) and structural (*e.g.* system hierarchy) information using a matrix metaphor. ExtraVis [2] displays the same dependency and structure data using a bundled edges metaphor. Finally, GemX [4] displays code duplication.

3. The Case Study

We used SQuAVisiT to study a large COBOL legacy system of a large insurance fund: 3 thousand modules, 1.7 million LOCs. The system provides support for a variety of online requests and batch executions. The system is at the starting point of migration to a new, presumably Java-based business rules engine platform. At this point it is crucial to assess the existing system to decide which system artifacts, *e.g.* architecture, business rules or code, are reusable and which are not, and what is the migration effort. We used SQuAVisiT to parse, extract metrics and call relations from the entire system in under 30 minutes on an ordinary PC. Figure 1 shows the metrics using the table lens technique: Every horizontal pixel line shows a different module, and

0-7695-2880-5/07 $25.00 © 2007 IEEE
DOI 10.1109/SCAM.2007.16

every column a different metric. The modules are sorted by descending McCabe metric value. This helped us locating the most complex (top) modules, which were less than 5% of the system.

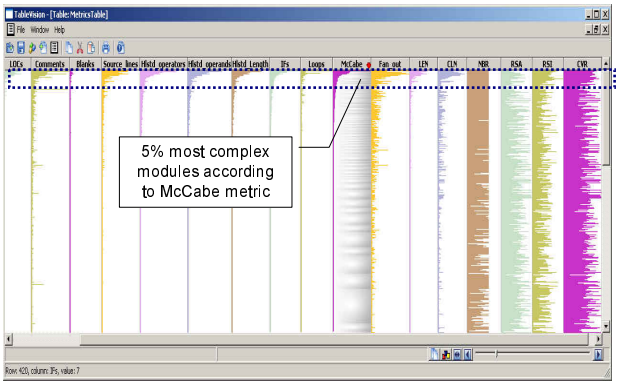

Figure 1. Metric visualization. Modules are sorted by descending McCabe metric value

Figure 2 shows the system structure, displayed as a set of concentric rings (modules, units, layers from inside to outside) and the calls as hierarchically bundled curves. This helped us to discover calls from a high-level module to the 'Business rules' layer. This violates a main design decision saying that business rules can be called from low levels only in order to make the system more maintainable.

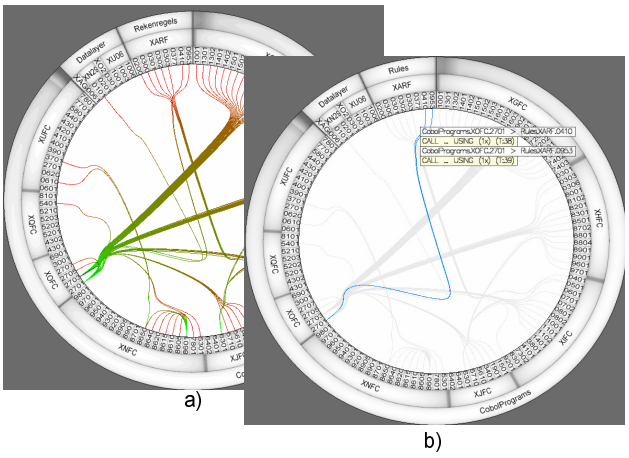

Figure 2. Call graph visualisation: a) Partial view on an execution subtree; b) Unexpected calls found: high-level module of XOFC unit (request handler) calls two 'Business rules' layer modules

Visualization with Gemx (Fig. 3) showed widely spread code duplication, especially at the 'Data' and 'Business rules' layers. Manual code inspection has shown unappropriate business rules coding style. This makes the system hard to maintain. Code reuse during migration will be possible only after necessary refinement.

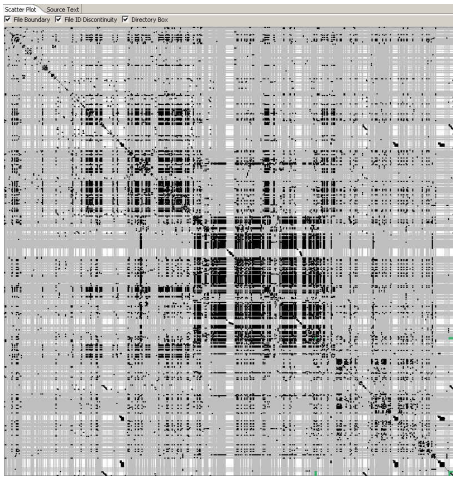

Figure 3. Scatter plot of the 'Business rules' layer. Widely spread black spots show code duplication

4. Conclusions

Our main contribution underlines the potential of reusing existing parsing, fact extraction, and visualization tools to develop a versatile analysis framework supporting code understanding for migration of large systems. Even though we used only simple metrics and extracted facts, the visualization helped putting these in perspective and gaining useful insights. We plan to extend our toolset with support for C/C++ and more visualization methods.

References

[1] J. Abello and F. van Ham. Matrixzoom: A visual interface for semi-external graphs. In *Proc. InfoVis*, pages 183–190, 2004.

[2] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741 – 748, 2006.

[3] JavaCC. Home page, *https://javacc.dev.java.net*, 2007.

[4] T. Kamyia. CCfinder home page, *http://www.ccfinder.net*, 2007.

[5] A. Telea. Combining extended table lens and treemap techniques for visualizing tabular data. In *Proc. IEEE EuroVis*, pages 51–58, 2006.