

Recovering management information from source code – Extended Abstract

Peter Kampstra

Lukasz Kwiatkowski

Vrije Universiteit Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

{pkampst, lukasz}@few.vu.nl

Keywords: source code analysis, management information, Cobol, software portfolio, IT-portfolio analysis, IT-portfolio management, software asset management.

At large software-empowered companies, such as banks, it can be a hard task to gain insight into the available source code. With over 50 million lines of code, it is no longer feasible to provide good insight in a manual fashion. Nevertheless, valuable management information often resides in the source code; for example, it is useful to know which error-prone modules ought to be rewritten. In this paper we therefore present some source code measures that can be extracted from millions of lines of code in an automated fashion. Source code comments usually contain data such as version history, module creation date and time of compilation. Interrelations between modules, the amount of code reuse and number of lines of code can be obtained by deploying relatively simple lexical scripting (e.g. Perl). From the available data we recover useful management information as described below. Source-derived information complements other available management information. Note that coupling the source-derived information to existing management information can be a difficult task. Of course, when other management information lacks, which is often the case with legacy systems, our approach is more a creator of such information. Such information is crucial to support decision making with relation to the evolution strategy of the software portfolio.

We take the following steps to create management information.

- Recovering the IT-portfolio size

Size is an important facet of software assets. While it is easy to measure the source lines of code (SLOC), function points are to be preferred (e.g.[2]). Sizing legacy systems requires access to the functional documentation, which in many cases is either missing or obsolete. A method that enables sizing by performing source code analysis would allow the source code to become a self-specifying entity; easily quantifiable and manageable. In Figure 1, however, we still use (physical) SLOC as the bases for showing the project sizes

encountered in a portfolio. Note that we used a project size index for confidentiality reasons. The empirical distribution shows that the majority of the software systems is relatively small, but that there are substantial outliers, which represent large and thus difficult-to-replace systems. Note that we think the example graph presents a normal or even good situation (as seen in [3]). If there are lots of large projects, there will probably be many more future problems.

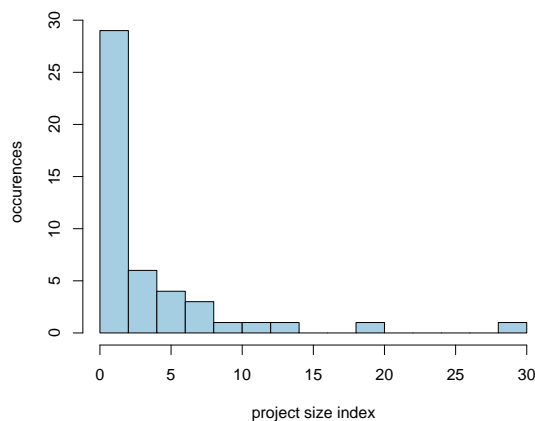


Figure 1: Portfolio project size-index distribution

- Identifying dead code

Large information systems usually contain code that is not used anymore. By identifying dead code, it becomes easier to assure that no money will be spent on its maintenance. This aspect is important when, for example, paying per code volume in outsourcing contracts. But also productivity of maintenance is lower for larger systems, so decreasing their size is key.

- Identifying points needing change

Sometimes it is necessary to identify where in the source code certain data structures are handled. The most well-known examples are Y2K and Euro, but there are many similar projects. In [1] an example estimation is done for a project in which bank account numbers were to be changed from 9 to 10 digits. In this project, by automating

detection, expected costs of change were reduced by a factor of 4.

- Identifying error-prone modules

From the source code, modules can be identified that need many bug fixes, for example because they were made on Monday morning. Management can then decide to rewrite these modules, in order to reduce costs.

- Recovering the ownership architecture

When scanning the source code comments an abundance of knowledge can be found; usually it is tagged by code owner's name. For enhancement projects the strategy to renovate depends on the availability of domain experts. Figure 2 presents how source code knowledge of a certain project is distributed among code owners. A footprint of the ownership architecture helps in defining the right strategy.

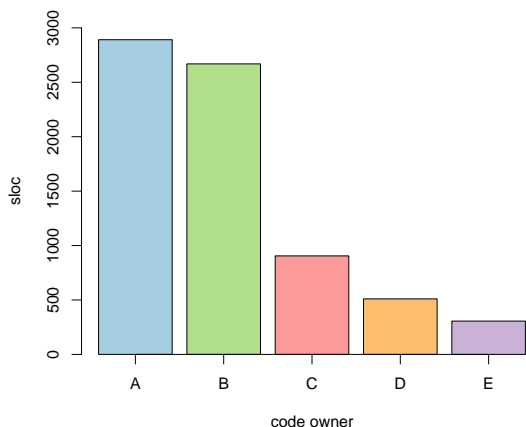


Figure 2: Project contribution per code owner

- Measuring code reuse

Modules that are used in many projects are very likely the most well tested components. Therefore, identification of these components is a good idea.

- Measuring source code volatility

Software volatility provides a good insight into the changeable nature of software. Knowledge of the volatility measure could help to better understand an organization's software evolution as well as improve its software engineering and systems management.

- Portfolio age

Inside large organizations legacy source code is still in use. When facing enhancement projects on legacy source code many difficulties are usually encountered, such as lack of suitable parsing technologies or insufficient expertise. By using

portfolio age information, common problems associated to an aging IT portfolio can be inventoried, so that proper action can follow. In Figure 3, we show portfolio age, measured in terms of latest date of compilation, for a healthy portfolio. An unhealthy portfolio would have lots of source lines which have not been compiled for many years.

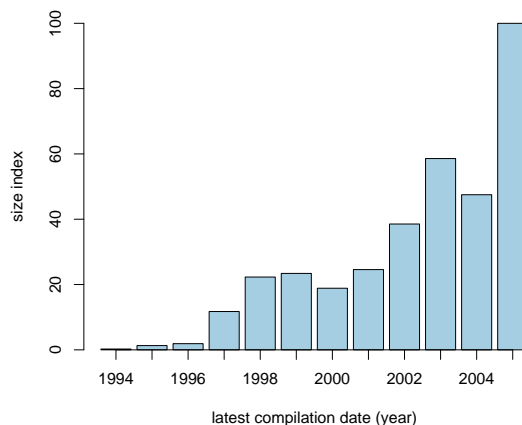


Figure 3: Module age

The above describes facets of source code analysis that we consider to be valuable from a managerial perspective. Other interesting future endeavors are:

- Measuring code quality
- Measuring architectural quality

In summary, an abundance of management information is hidden inside millions of lines of code. Because sources are the ultimate reality of an organization, it is crucial to reveal their management information. Source code analysis can be (almost) fully automated and therefore be made low cost, while it can provide invaluable management information. Business value can therefore be created by the extraction of management information through software re-engineering techniques.

Acknowledgements This research was sponsored by the Dutch Joint Academic and Commercial Quality Research & Development (Jacquard) program on Software Engineering Research via contract 638.004.405 *EQUITY: Exploring Quantifiable Information Technology Yields*.

References

- [1] A. Klusener and C. Verhoef. 9210: The zip code of another IT-soap. *Software Quality Journal*, 12(4):297–309, Dec. 2004.
- [2] C. Verhoef. Quantitative IT portfolio management. *Science of Computer Programming*, 45(1):1–96, 2002.
- [3] C. Verhoef. Quantifying Software Process Improvement, 2004. Available via www.cs.vu.nl/~x/spi/spi.pdf.