Recovering management information from source code – Extended Abstract

Peter Kampstra
Łukasz 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, a good insight into millions of lines of source code is often missing. Nevertheless, valuable management information often resides in the source code. 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. Coupling source-derived information to existing management information is not trivial, but enables a check for correctness of the existing management information. Source-derived information also complements existing management information. Of course, when there is no existing management information, which is often the case with legacy systems, our approach creates such information. This information is crucial to support decision making with relation to the evolution strategy of the software portfolio.

We take the following steps to do a periodic source-code health check.

- Recovering the IT-portfolio size

  Size is an important facet of software assets; for example, size is used in software cost estimation models. 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 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. A project size index is used 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 systems, there will probably be many more future problems.

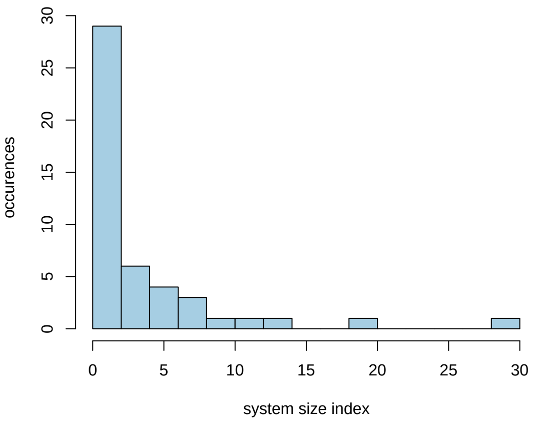

Figure 1: Portfolio system size-index distribution

- Measuring source code volatility

  Volatility metrics help us to understand an organization's software engineering and systems management. A higher volatility can be an indication of higher maintainance costs, lower reliability and/or need for more design.

- Identifying error-prone modules

  High volatility and complexity can indicate modules that need many bug fixes, for example because they were written on Friday afternoon. Management can then decide to rewrite these modules, in order to reduce costs.

- 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 2, we show portfolio age, measured in terms of latest date of compilation, for a healty portfolio. A portfolio with lots of source lines that have not been compiled for many years is associated with higher risks once maintenance is needed.

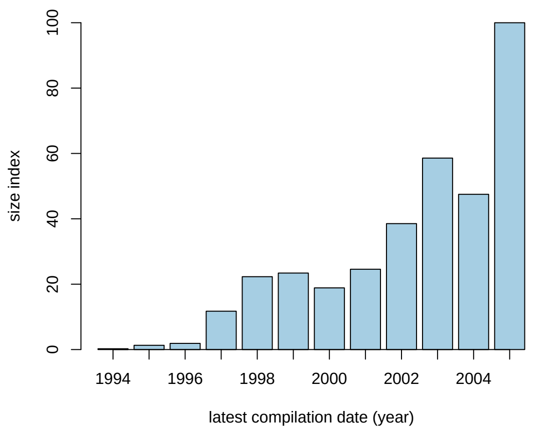

Figure 2: Module age

Apart from size, volatility, age and complexity data for (cost) estimation, the following source-derived information can support decision making in new (maintenance) projects.

- 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.

- 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.

- 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.

- 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 3 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.

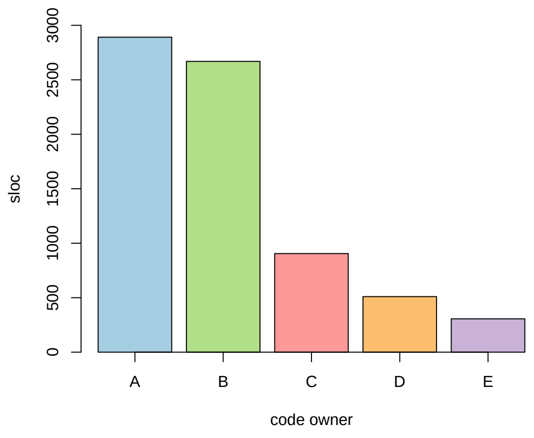

Figure 3: Project contribution per code owner

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 reengineering 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`.