A Case Study on the Transformation From Proprietary to Open Source Software

Alma Oručević-Alagić and Martin Höst

Department of Computer Science, Lund University, Sweden,
Alma.Orucevic-Alagic@cs.lth.se,Martin.Host@cs.lth.se

Abstract. This paper presents an extensive analysis of static software quality metrics changes for an open source enterprise database management system (DBMS), as the software was moved from the proprietary into open source software development environment. The software quality metrics of special interest for the research are cyclomatic complexity, effective lines of code, the degree of system modularity, and the amount of comments in the code.

1 Introduction

Popularization of OSS has influenced the conventional way in which companies perceive commercial value of software. The companies recognized that commercial value of product can come from other sources rather than conventional sale of software licenses. This research assesses the impact of source code changes made by OSS community to software that was transitioned from proprietary into open source, in terms of static software quality metrics. The case software analyzed is the Ingres database management system (DBMS) [2], which, according to many, has received a new breath of life after its release into the open source community.

2 Background

Stemlos [3] conducted code quality analysis in open source development for 100 applications written for Linux. It was determined that some open source products have lower quality of code produced in OSS environment then that which is expected as an industry standard.

The very roots of the case software reach back to the 1970s and UC Barkley, when the initial development of the software was started as open source. The same code base was modified and spawned into Sybase and Microsoft SQL server in 1980s. In 1994, the software was acquired by CA (Computer Associates) from the ASK Group, the company that created a proprietary version of the Ingres code. In order to increase the market share, CA decided to transform the product to open source in 2004, by implementing loss-leader/market positioner business model [4]. In November of 2005, Computer Associates and Garnett and

Helfirch capital created a new company, Ingres Corporation. The main role of Ingres Corporation is to oversee the open source development process. Today, Ingres customer base includes 10,000 enterprise customers, among which 136 belong to the Fortune 500 companies like 3M, Bea Systems, and Lufthansa [6].

The high level architecture of the case software is grouped into four major components:

Front End: Functionality covers user interface facilities.
Back End: Functionality covers DBMS server functionality.
Common: Functionality covers connectivity and communications between the front end the back end.
Utility: Functionality covers utility libraries that interact with operating system.

3 Research Approach

The following research questions were investigated during the research:

1. What parts of the Ingres DBMS software components went through the most source code changes in terms of source files added, changed, and deleted?
2. How did Ingres DBMS code base change under the OS community process in terms of static source code metrics?

It is important to highlight the objective of this study, i.e. to understand what changes that have been carried out, and not to assess or compare the case software to any other software. The study is conducted as a case study [5].

3.1 Data collection

In order to analyze and compare code metrics of the most recent proprietary version, further referred as 2004v, and open source version, further referred as 2008v, of Ingres, the 2004v was obtained by directly contacting the Ingres Corporation. The 2008v was downloaded from the Ingres Open Source community web site in November of 2008.

A program that parses through the 2004v and 2008v code base was created, or more specifically, the files and subdirectories under the main `src` directory that contains all of the source files. The files were compared between the two code bases in order classify all files according to the following:

- File type 0 : Source files that can be found only in 2004v
- File type 1 : Source files identical - unchanged between the 2004v and 2008v
- File type 2: Source files that were changed between the 2004v and 2008v
- File type 3: Source files that were added in 2008v

Metrics were measured in both versions are: lines of code (LOC), effective lines of code ($ELOC$), comment lines (C), total cyclomatic complexity (TCC), and file functions count (FFC). All metrics are calculated on file level.

For coding purposes developers often use braces or parenthesis to make code more readable, but this practice can inflate LOC metrics [7]. The $ELOC$ metric takes into consideration all lines of code except blank only or comment only lines as well as the lines containing only standalone braces or parenthesis (`{`, `}`, `(`, `)`) Thus, lines counted by the $ELOC$ metric are a subset of the lines counted by the LOC metric.

C denotes the number of comment lines. The comment lines can appear by themselves on one physical line of code, or can be co-mingled.

The TCC or total cyclomatic complexity metric, also known as McCabe's cyclomatic complexity, is the degree of logical branching per source file.

FFC, or total number of file functions, within a source file determines the modularity of the file. The FFC metric combined with $ELOC$ metric produces average number of effective lines of code, $AELOC = ELOC/FFC$. In the same way, the average cyclomatic complexity is calculated as $ACC = TCC/FFC$.

In addition to the above metrics, the amount of comments are of interest. Therefore a metric describing the relative number of comments in each file is calculated as $RC = C/(ELOC + C)$.

Metrics for each file were derived with a metrics tool and stored in a database together with file type information for analysis.

3.2 Analysis

Analysis with respect to research question 1 was conducted by determining the percentage changes in terms of file type 0, file type 1, file type 2, and file type 3 per major components of the source code.

When analysing research question 2, the differences between the different versions of the case software were investigated with hypothesis tests. The null hypotheses state that the code changes made to 2004v, resulting in 2008v, had no impact on code metrics.

Let $T = \{0, 1, 2, 3\}$ denote file types according to above and let $M = \{AELOC, ACC, RC\}$ denote the different metrics of interest, so that $\mu_m(v, T_s)$ represents the expected mean of metric $m \in M$ for all files of types $T_s \subseteq T$ in version v. Then the following null hypotheses have been defined:

$$\begin{aligned} H0_{m,changed} &: \mu_m(2004v, \{2\}) = \mu_m(2008v, \{2\}) \\ H0_{m,new} &: \mu_m(2004v, \{0, 1, 2\}) = \mu_m(2008v, \{3\}) \\ H0_{m,all} &: \mu_m(2004v, \{0, 1, 2\}) = \mu_m(2008v, \{1, 2, 3\}) \end{aligned}$$

That is, three null hypotheses have been formulated for each metric in M so that there is one concerning only the changed files ($H0_{m,changed}$), one concerning all files from 2004v and only newly added files to 2008v ($H0_{m,new}$), and one concerning all the files in 2004v and 2008v ($H0_{m,all}$). This means that $|M| \times 3 = 3 \times 3 = 9$ null hypotheses and equally many alternative hypotheses have been defined in total.

Analysis of data for distributions of metrics results for version 2004v and 2008v were performed and it was determined that data for the metrics do not follow normal distribution. Hence in order to compare distribution of the metrics, non parametric tests, Mann-Whitney and Wilcoxon were performed. The Wilcoxon Signed-Rank Test for matched pairs was used in order to compare paired data sets (i.e., in analysis of $H0_{m,changed}$), and the Mann-Whitney U test was used to compare un-paired data (i.e., in analysis of $H0_{m,all}$ and $H0_{m,new}$).

4 Results

Not all of the source code subdirectories will be analyzed in more detail, but only front, back, common, gl, and cl, since these directories contain almost 95% of the code. Hence, the most of the source files are located under /src/front directory or 54.7% of all 2008v. In the second place is src/cl directory housing 15.40% of source files in 2008v, followed by the src/back and src/common, housing 14.06% and 10.44% of all 2008v source files, respectively. Thus, these four directories contain 94.6% of 2008v source files.

Under the src/front directory the components that belong to the front end layer of the software are stored. Over 50% of all changes in the front end layer are due to the addition of the new source file components (type 3). Another 33% of changes are due to changes (file type 2). Thus, around 88% of front end source files have been changed since the case software went open source. Under src/cl library source files for Ingres Compatibility Library are housed. This library grew 69% between 2004v and 2008v, that is it contains 69% of file type 3 files. The src/back end components are deemed very important as the proper functioning of these components significantly affects database performance. The back end components went through the least amount of source code changes and additions, having 67.5% of code unchanged (file type 1) between the 2004v and 2008v. It also contains the least number of file additions (file type 3), thus only having 2.92% of the total number of the source files added (file type 3). Finally the src/common contains components used by both, front and back end. The common components contain 49.05% of file type 1, or almost half of its components are same for 2004v and 2008v. It can be observed that 19.5% of its file were of file type 3, or newly added components.

Table 1 displays code metric statistics summarized for the entire source code base of 2004v and 2008v. Hence, it can be observed that the number of file functions, lines of code and effective lines of code has increased. As one would expect, the higher number of functions and lines of code produced higher values for total cyclomatic complexity of 2008v code compared to 2004v.

The results of hypothesis testing for the stated hypotheses are presented in Table 2 (significance level 0.05).

Table 1. Summary of source code metrics for the whole system

Code Metric	2004v	2008v
Total LOC	840,502	1,442,225
Total $ELOC$	650,055	1,110,261
Total C	484,349	630,635
Total TCC	167,753	300,493
Total FFC	15,588	45,216

Table 2. Mean values and results of hypothesis tests

$H0$	mean 2004	mean 2008	p	reject $H0$
$H0_{AELOC,changed}$	41.35	41.69	< 0.001	yes
$H0_{ACC,changed}$	10.47	10.80	< 0.001	yes
$H0_{RC,changed}$	0.53	0.54	1	no
$H0_{AELOC,new}$	23.68	11.85	0.0042	yes
$H0_{ACC,new}$	6.12	2.80	0.01	yes
$H0_{RC,new}$	0.56	0.42	< 0.001	yes
$H0_{AELOC,all}$	23.68	19.02	0.1383	no
$H0_{ACC,all}$	6.12	4.85	0.1841	no
$H0_{RC,all}$	0.56	0.50	< 0.001	yes

Concerning $AELOC$, this metric is somewhat increased for changed files, meaning that when files are changed the functions in the files have become somewhat larger. For new files the metric is much lower than for old files, meaning that functions in new files are smaller than in older files. In total, looking at all files, the metric is higher in the new version than in the older version. The differences are statistically different for changed code and new code compared to old code, but not for all code.

Concerning ACC the same type of observation as for $AELOC$ can be made. For changed files the complexity is slightly higher and for new files the complexity is much lower. For RC there is no significant difference for changed code, but for new code there are significantly less comments. In total there is relatively less comments in the new version compared to the old version.

5 Conclusions

The conducted analysis have shown that over half of the changes made to the case source code were made in the front end group of source code components, while the least of the changes were seen in the back end components. There can be many reasons for this, e.g. simply that more changes were needed in these components, but another reason may be that these are nearer to the interest of the new community that was formed during the open source transition process.

The results of comparison of code quality metrics between all files in 2004v and new files in 2008v show significant and large decrease in ACC and $AELOC$, that is, significant and large increase in quality metrics for code developed by the OSS community. The code quality decrease in metrics smaller than the increase of the changed files, and as a result the code quality metrics for 2008v are higher than those of the 2004v, but this increase in code quality is not significant. This means that the overall code quality metrics, in terms of average cyclomatic complexity and the average effective lines of code per function has increased somewhat for changed code, and decreased rather much for new code. This can be interpreted as an improvement for added code. The number of comment lines per effective lines of code ACC has decrased and there are significantly less comments in newly added code. At the same time the number of comments per effective lines of code (RC) has seen significant decrease between the 2004v and 2008v of source code base. Hence, while there was a small improvement in ACC and $AELOC$, the lower number of comments per effective lines of code suggests that code in OSS community was not documented as much as in closed source environment.

The transition of the software was also accompanied by 100% increase in customer base, out of which some 138 customers belong to the Fortune 500 group, and 32% revenue increase reported for the 2008.

For companies planning to go open source, this study can provide an example on how the OSS community can impact static software quality metrics.

Acknowledgments

The authors would like to express their gratitude to the Ingres Corporation for providing us with a last proprietary version of the software. This work was partly funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering, (`http://ease.cs.lth.se`).

References

1. Norman E Fenton and Shari Lawrence Pfleeger. Software Metrics: A Rigorous and Practical Approach, Revised. PWS Publishing Company, ITP International Thomson Publishing Company(1998)
2. IngresWebSite. Official web site of ingres corporation. http://ingres.com/ (2009)
3. Apostolos Oikonomou Ioannis Stamelos, Lefteris Angelis. Code quality analysis in open source development. Information Systems Journal, 12(1):43–60(2002).
4. Eric S. Raymond. The Cathedral and the Baazar O'Reilly Media, Inc. (2001)
5. Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14:131–164 (2008)
6. Matt Assay. February 2009 web server survey. `http://news.cnet.com/8301-13505_3-10156188-16.html`(2009)
7. RSM Effective lines of code eloc metrics for popular open source software linux kernel 2.6.17, firefox, apache hppd, mysql, php using rsm. `http://msquaredtechnologies.com/m2rsm/docs/rsm_metrics_narration.htm` (2008)