Examining the Evolution of Code Comments in PostgreSQL

Zhen Ming Jiang and Ahmed E. Hassan
Software Architecture Group (SWAG)
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
{zmjiang, aeehassa}@uwaterloo.ca

ABSTRACT
It is common, especially in large software systems, for developers to change code without updating its associated comments due to their unfamiliarity with the code or due to time constraints. This is a potential problem since outdated comments may confuse or mislead developers who perform future development. Using data recovered from CVS, we study the evolution of code comments in the PostgreSQL project. Our study reveals that over time the percentage of commented functions remains constant except for early fluctuation due to the commenting style of a particular active developer.

Categories and Subject Descriptors
D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement – *Documentation.*

General Terms
Human Factors.

Keywords
Software Evolution, Software Maintenance, Code comments.

1. INTRODUCTION
Most of the software development effort is devoted to software maintenance. Developers spend about half of their time trying to understand code [1]. Most developers agree that it is not easy to read other people's code. A well documented program is easy to follow and improves the quality of the software [3]. However, in large software systems, due to unfamiliarity with the system or due to time constraints or maybe just laziness, developers are likely to change source code without updating its associated comments. This is a potential time bomb, since outdated comments are misleading and cause confusion. We believe it is worthwhile for managers to monitor the evolution of code comments over time.

We study source code comments in the PostgreSQL project over time. Our focus is on the comments associated with functions. We categorize code comments into two types: *Header Comments* and *Non-Header Comments*. *Header Comments* are comments before the declaration of a function; whereas *Non-Header Comments* are all other comments residing in the body of a function or trailing the function. Developers usually use Header Comments to describe the purpose of a function, and to document its parameters and interfaces. Non-Header Comments are usually used to document algorithms and low level design decisions.

MSR'06, May 22–23, 2006, Shanghai, China.
ACM 1-59593-085-X/06/0005.

Research by Perry *et al*. has shown that at least 66% of bugs in large projects are due to interface errors [4]. Uncommented interfaces or interfaces with outdated comments are likely to cause bugs. In this paper, we examine whether the percentage of functions with header comments (FH) drops over time relative to the functions with non-header comments (FNH). We believe that a drop may indicate that developers are not updating the interface documentations.

2. DISCUSSION ABOUT OUR FINDINGS
To perform our study, we used the C-REX extractor [2] to recover all CVS changes for PostgreSQL from 1996 to 2005. C-REX is able to track the addition and removal of functions and function dependencies over time. It also tracks all changes to comments associated with these functions.

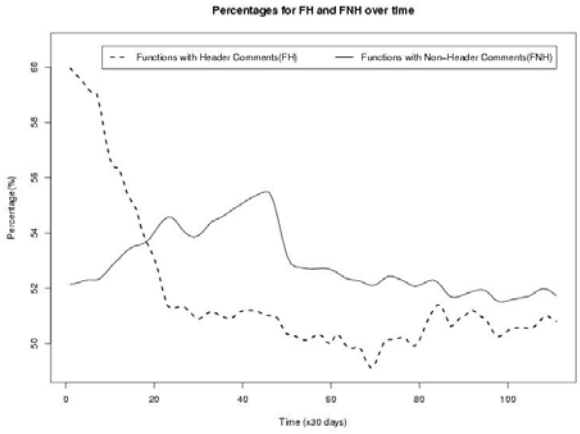

Figure 1: Percentage of FH and FNH Over Time.

Figure 1 shows the percentage of functions with header comments (*FH*) and non-header comments (*FNH*) for every 30 days period. This Figure reveals that:

1. During the initial two year period (the first 30*25 days), there is a steady decrease in the percentage of FH and an increase in the percentage of FNH.
2. After the initial two year period, the percentage of FH and FNH remain steady, and are around 51% and 52%, respectively.

The first finding is worth investigating since it may be due to the removal of many FH or the addition of a large amount of FNH relative to FH. It is also possible that quite a few FH had their header comments removed. The addition of many FNH is troublesome since the interfaces for these functions are not documented and may cause future bugs.

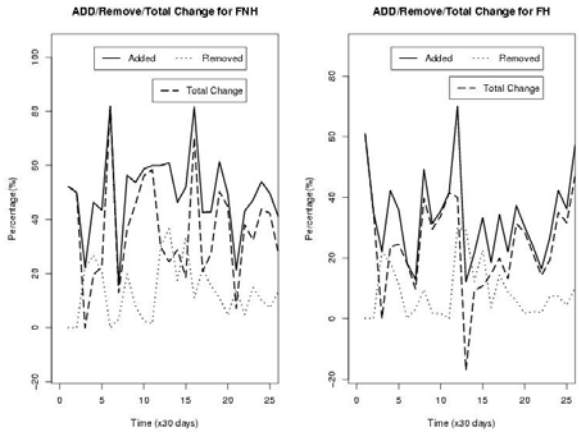

Figure 2: Percentage of Added/Removed/Total Change for FNH and FH.

To investigate the changes in the percentage of FH and FNH, we plot the percentage of addition and removal of FH and FNH during the first two year period in Figure 2. The Figure as well shows the total change (percentage of added – percentage of removed) over time. In both subgraphs in the Figure, we note that the total change line is always above zero (except one case around 13 in the right subgraph). Therefore, we can conclude that more FNH and FH are added than removed during this two year period.

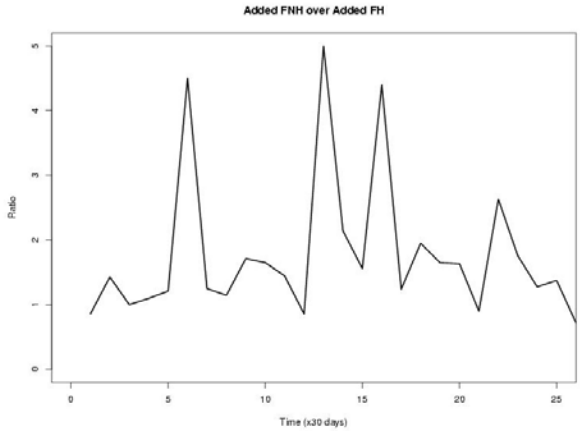

Figure 3: Ratio of Number of Added FNH over Added FH

We now compare the amount of added FNH against the amount of added FH. Figure 3 shows the relationship between added FNH and FH for the two year period. The Figure plots the ratio of added FNH over added FH for every 30 days. We see that the ratio always stays above 1. This indicates that there are always more FNH being added than FH during the initial two year period.

Using the recovered C-REX data which tracks all changes to the source code and the name of the developers who performed these changes, we examine closely the spikes in Figure 3. Our investigation reveals that these spikes are due to a particular developer who contributed a large number transactions during these time periods. These transactions added mainly utility functions to PostgreSQL. The developer has a particular commenting style, where he appends the name of a function at the end of the function's declaration block. For example, in revision 1.13 of the file "./postgres/pgsql/src/backend/utils/adt/geoops.c", he adds a small uncommented utility function called "int4 text".

```
text * int4_text ( int32 arg1 )
{
. . .
} /*  int4_text ( )   */
```

If this method were added by other developers, it would probably become a function with no comments at all; however, in this case it belongs to the category of FNH functions.

3. CONCLUSION AND FUTUREWORK

Correct and up to date comments aid developers in understanding the source code; wrong or outdated comments mislead developers and cause the introduction of bugs. Thus, it is important that managers monitor code comments over time. In this paper, we studied comments in PostgreSQL. We discovered that apart from the initial fluctuation due to the introduction of a new commenting style; the percentage of functions with header and non-header comments remains consistent throughout the development history.

In the future, we plan to investigate the relationship between the decrease in comment rate and the introduction of bugs.

4. REFERENCES

[1] R. Fjeldstad and W. Hamlen. Application program maintenance-report to our respondents. In *Tutorial On Software Maintenance*, pages 13–27.1983.

[2] A. E. Hassan and R. C. Holt. C-REX: An Evolutionary Code Extractor for C. May 2004.

[3] D. Parnas. Software aging. In *Proceedings of the 16th International Conference on Software Engineering*, pages 279 – 287, Sorrento, Italy, May 1994.

[4] D. E. Perry and W. M. Evangelist. An Empirical Study of Software Interface Faults—An Update. In *Proceedings of the 20th Annual Hawaii International Conference on Systems Sciences*, pages 113–136, Hawaii, USA, Jan. 1987