

Do Comments Explain Code Adequately?

-Investigation by Text Filtering-

Yukinao Hirata, Osamu Mizuno
Kyoto Institute of Technology



Brief Summary

- We investigated in which projects (Eclipse and Netbeans) comment explains code adequately.
 - By probability of fault-proneness using text-filtering method for comment and code.
- We concluded that Eclipse has more adequate comment.

Introduction

- The comment in the program is used for coding and maintenance.
- The wrong description in comments may cause further bugs and confusion in maintenance.
- A method to investigate whether comments are adequate is necessary.

Fault-prone Filtering

- Fault-prone filtering [1] is a fault-prone module detection method inspired by spam mail filtering.



[1] O. Mizuno and T. Kikuno. Training on errors experiment to detect fault-prone software modules by spam filter. In Proc. of 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering, 4 pages 405–414, 2007.

Definition of Comment lines

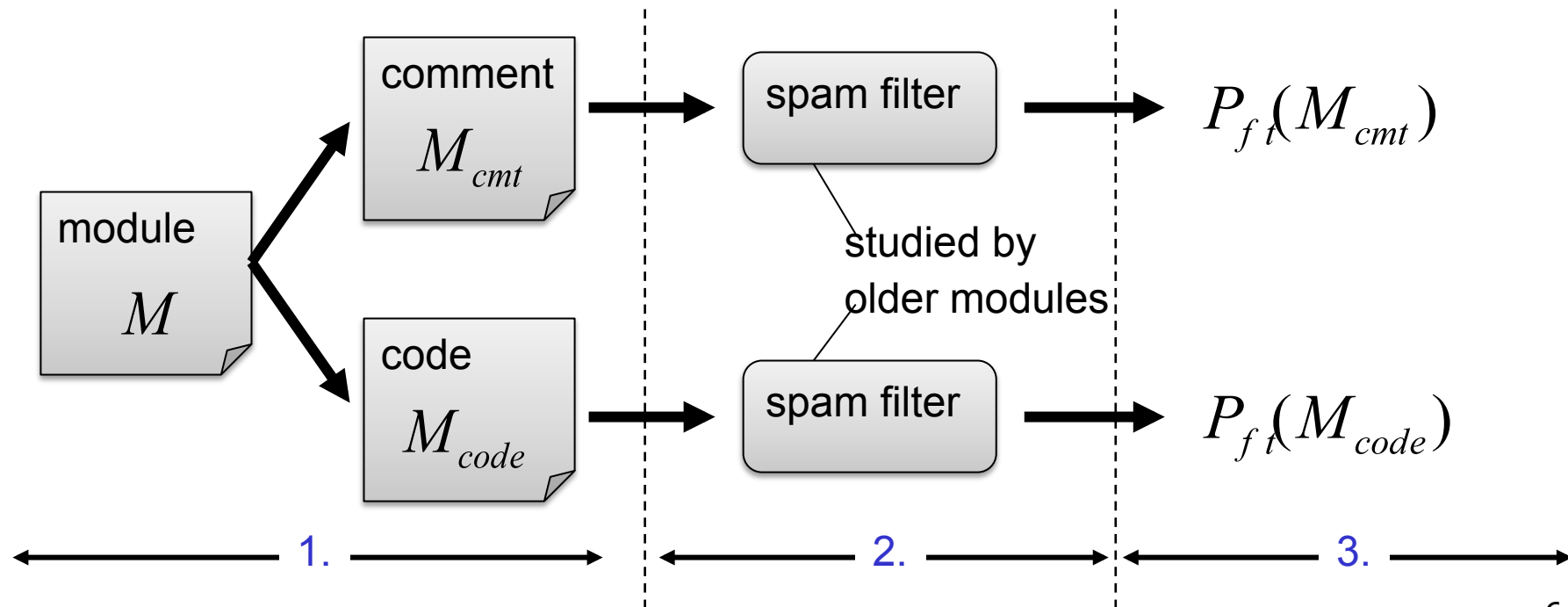
- There are three types of comments in Java.
 - `/* comment */`
 - `/** comment */`
 - `// comment`

In this research ...

Comment line includes all types of comments

Experiments

1. Extract comment and code from module
2. Apply comment and code through spam filter
3. Calculate distance between code and comment



Distance between Code and Comment

- A distance between code and comment is defined as follow:

$$D(M) = | P_{ft}(M_{cmt}) - P_{ft}(M_{code}) |$$

M : module

P_{ft} : probability of faulty

M_{cmt} : part of comment in module M

M_{code} : part of code in module M

- We assume that $D(M)$ shows the degree of adequateness of the comment for code.

Target Projects

Table 1: The group 1 of mining challenge

Project	Eclipse	Netbeans
Number of modules	308,966	393,531
Number of faulty modules	159,818 (51.7%)	165,560 (42.1%)
Number of non-faulty modules	149,148 (48.3%)	227,971 (57.9%)
Total lines	159,175,979	130,619,502
Comment lines	52,417,180 (32.9%)	36,272,795 (27.8%)
Code lines	106,758,799 (67.1%)	94,346,707 (72.2%) ⁸

Results

Table 2: Prediction Results in Eclipse and Netbeans

	Eclipse			Netbeans		
	Accuracy	Precision	Recall	Accuracy	Precision	Recall
Comments	0.651	0.947	0.345	0.658	0.944	0.199
Code	0.838	0.927	0.746	0.857	0.857	0.857

Table 3: Average Distance in Eclipse and Netbeans

	Eclipse		Netbeans	
	Correspondence	Average D^2	Correspondence	Average D^2
Comments vs Code	0.738	0.256	0.707	0.290

Conclusions

- Comments in Eclipse is more adequately described than in Netbeans

The end

