

オープンソースソフトウェアにおける コメント記述及びコメントアウトと フォールト潜在との関係に関する定量分析

阿 萬 裕 久^{†1}

一般にコメント文はソースコードを理解する上で有益な情報を提供する。しかしながら、理解性の低いコードや複雑なコードであってもコメント文の存在によってそれらの問題が目立たなくなる場合もある。リファクタリングの分野では、そういったコードの問題を“不吉な匂い”と呼んでおり、その場合にコメント文は“消臭剤”の役割を成すともいわれている。そこで、実際の傾向について考察するため、本論文では七種類のオープンソースソフトウェアに対してデータ収集と定量分析を行っている。その結果として次の傾向が確認されている：(1) コメント記述の多いコードほどフォールトの潜在率も高い傾向にある；(2) LOC に対するコメント記述の割合が 12.6% を超えるコードでのフォールト潜在率は、コメント記述の無いコードの約 2 倍である；(3) コメントアウトも併せて登場するコードではその可能性が更に高い。

Quantitative Analysis of Relationships among Comment Description, Comment Out and Fault-Proneness in Open Source Software

HIROHISA AMAN^{†1}

Comment descriptions generally provide useful information for understanding the source code. However, some comments mask problematic code fragments which are complex and/or hard-to-understand. In the study of refactoring, such problematic code fragments are called “bad smell” and the associated comments serve as “air freshener.” To investigate such impact of comments, this paper collects a lot of source code from seven major open source software and conducts quantitative analysis on their source code. The results show the followings: (1) A code having more comments is more fault-prone; (2) A code, whose ratio of comments to LOC is greater than 12.6%, is about double fault-prone than a code which has no comments; (3) The assurance of the above trend is higher when the code has not only many comments but also a comment-out.

1. はじめに

ソースコードにおいて、コメント文はプログラムの実行に一切影響しない要素である。しかし、コメント文の記述内容は当該ソースコードの内容説明である場合や開発者のメモである場合もあり、そのコードの理解容易性には大きな影響力を持つといえる。それゆえ一般に、コメント文の記述はコードレベルでの重要な品質維持・向上活動として認識されている。実際、コーディング規則の一部としてコメント文記述に関する規則^{1),2)}が設けられている場合も多い。

しかしその一方で、コメント文が多く書かれたコードに対する警鐘も見られる。Fowler³⁾は、リファクタリングでいうところの“不吉な匂いの予兆”としてコメント文を挙げている。そこではコメント文そのものを非難しているのではなく、それが“消臭剤”の役割になってしまい、結果として不吉な匂いのコードを隠してしまっているのではないかと指摘されている。また、Kernighan ら⁴⁾はプログラミング作法として、多くのコメントを書かなければならないのであれば、むしろそのプログラムを書き直すべきであると述べている。端的に言えば、コメント文は品質を低下させるような一種の“悪い存在”というわけではない。しかしながら、結果としてコメント文が多く書かれるようなコードは、そのコメント文が無ければ理解できないようなコードになっており、それゆえフォールトの温床になっている（質の低下を招いている）のではないかという懸念がある。

そこで本論文では、実際のオープンソースソフトウェアに対してコメント文記述の状況を調査し、フォールトの潜在との関係について定量的な分析を行う。分析を行うにあたってオープンソースソフトウェアを調査対象とした理由は、オープンソース開発のようにソースコードが公開される環境では、企業等の内部に閉じた開発環境よりも上述の傾向が顕在化しやすいと考えたためである。また本論文では、コメント文の記述のみならずコメントアウトについても分析を行う。ソースコードが公開されており、なおかつリポジトリを介して過去のコード改変も追跡可能であるという環境において、敢えて削除することなく過去のコード断片をコメントのかたちで残していることにも何らかの意味があると考えられるため、コメントアウトも調査・分析の対象とした。

^{†1} 愛媛大学大学院理工学研究科

Graduate School of Science and Engineering, Ehime University

2 オープンソースソフトウェアにおけるコメント記述及びコメントアウトとフォールト潜在との関係に関する定量分析

以下, 2 節ではコメント文記述, コメントアウト及びフォールト潜在の状況を定量的にとらえるための準備として, それら三つの視点に関するメトリクスを導入する. そして, 3 節において各メトリクスによる調査結果を示し, その上で実際のフォールト潜在との関係について定量的な分析を行う. 4 節で関連研究についてふれ, 最後に 5 節で本論文のまとめと今後の課題について述べる.

2. メトリクス

本論文では, ソースコードにおけるコメント記述の様子を定量的にとらえるため, 次のメトリクスを導入する. 便宜上, 対象は Java 言語で書かれたソースコードとする.

- コメント記述率 (CoMment Ratio: CMR)⁵⁾,
- コメントアウト率 (Comment OUT Ratio: COUTR)⁵⁾.

定義 1 (コメント記述率 (CMR))

ソースファイルが与えられたとき, CMR 値を (1) 式で定義する:

$$\text{CMR} = \frac{\text{コメント行数}}{\text{LOC}}, \quad (1)$$

ただし, コメント行数とはコメント文が行全体あるいは一部に記述されているような行数である. なお, ここでいうコメント文はメソッド定義の内部に記述されたものであって, なおかつコメントアウトでないものに限る. また, LOC (Lines Of Code) は当該ソースファイルにおけるコメントのみの行と空行を除いたコード行数である.

□

CMR はソースファイルにおけるコメント文記述の量を規模 (LOC) に対する比率として表したメトリクスである. なお, ここで対象としているコメントは, メソッドの内部に記述されたものに限定されており, Javadoc のようなドキュメンテーションコメントはその目的が他のコメント文と異なることから対象外としている. 同様に, この測定ではコメントアウトも含まない. コメントアウトについては, 次のように別途メトリクスを導入する.

定義 2 (コメントアウト率 (COUTR))

ソースファイルが与えられたとき, COUTR 値を (2) 式で定義する:

$$\text{COUTR} = \frac{\text{コメントアウトの行数}}{\text{LOC}}. \quad (2)$$

ただし, コメントアウトの行数とは, コメントアウトが行全体あるいは一部において行われている行数である. なお, 与えられたコメントがコメントアウトであるかどうかの判別

は, 文献 5) のアルゴリズムによって行うものとする.

□

COUTR はソースファイルにおけるコメントアウトの量を規模 (LOC) に対する比率として表したメトリクスである. コメントアウトは通常のコメントと異なり, ソースコードの一部を無効化する目的で使用される. その際, 敢えて削除せずにコメントとして残すという行為にも一定の意味はあると筆者は考え, 前出のコメントとは区別して測定する.

本節の最後に 3 節での分析に向けた準備として, 次の “フォールト潜在率 (Fault Ratio: FR)”⁵⁾ も導入しておく.

定義 3 (フォールト潜在率 (FR))

ソースファイルの集合が与えられたとき, FR 値を (3) 式で定義する:

$$\text{FR} = \frac{\text{フォールトが検出されたソースファイル数}}{\text{対象ソースファイル数}}. \quad (3)$$

□

FR は, 与えられたソースファイル集合の中でリリース後に一つ以上のフォールトが検出されたソースファイルの占める割合である. つまり, 所定のソースファイル集合においてフォールトの潜在がどれだけ疑われるのか (Fault-prone であるか) を表したメトリクスである. 多くの Fault-prone モジュール予測研究と同様に⁶⁾ FR は各ソースファイルでのフォールトの有無に着目している (フォールト数ではない) ことに注意されたい.

なお, これに関しては, フォールトの個数や密度 (1KLOC あたりのフォールト数) といったメトリクスの使用も考えられる. 上述の FR はフォールトの有無に関するものであるため, 個数や密度に比べて情報が粗いという欠点がある. しかし本論文では, まずは “コメント文やコメントアウトがフォールトの潜在と関係しているかどうか” の解明を主たる研究課題と位置付け, FR を用いることとした. まずはフォールト潜在との関係の有無を明らかにし, その上で (今後の課題として) より詳細な観点であるフォールト数や密度との関係分析を行っていくべきであると考え, 今回は FR による分析を行うこととした.

3. 分 析

本節では, 前節で定義した三つのメトリクス (CMR, FR 及び COUTR) を用いてコメント記述とフォールト潜在の可能性との関係を定量的に分析する.

3.1 分析対象

分析対象のソフトウェアを表 1 に示す. 解析の都合上, 次の三点を満たしているという

3 オープンソースソフトウェアにおけるコメント記述及びコメントアウトとフォールト潜在との関係に関する定量分析

表 1 分析対象ソフトウェア
Table 1 Software used in our analysis.

ソフトウェア	バージョン	ソースファイル数	規模 (LOC)
Apache Ant ⁷⁾	1.3	122	14, 230
	1.4	173	20, 499
	1.5	286	33, 694
	1.6	343	45, 796
	1.7	720	87, 334
Apache log4j ⁸⁾	1.0	111	7, 636
	1.1	98	7, 180
	1.2	165	14, 004
Apache Tomcat ⁹⁾	6.0.0	758	122, 512
Apache Velocity ¹⁰⁾	1.4	184	22, 667
	1.5	206	24, 591
	1.6	219	26, 285
Apache Xalan ¹¹⁾	2.4	634	94, 971
	2.5	716	117, 692
	2.6	828	149, 328
	2.7	850	155, 094
Eclipse ¹²⁾	2.0	6, 143	765, 765
	2.1	4, 838	690, 736
	3.0	9, 727	1, 262, 893
Xerces ¹³⁾	1.2	411	60, 032
	1.3	425	64, 503
合計		27, 957	3, 787, 442

理由でこれらを分析対象とした：

- (1) ソースファイルが入手可能であること，
- (2) Java 言語で記述されていること，
- (3) 各ソースファイルに対し，そのリリース後でのフォールト検出の有無について情報を入手可能であること．

(1) 及び (2) は我々が開発したデータ収集ツールに起因する制約である。(3) は前述のコメント関連メトリクスとフォールト潜在率との関係を解析する目的に必要な制約である．本論文では (3) を満足するデータ集合として，PROMISE データリポジトリ¹⁴⁾ で公開されているフォールト検出情報を利用した．なお 表 1 では，実際に公開されているソースファイルのうち，PROMISE データリポジトリ上のフォールト情報と対応付けが可能なものに限定して掲載している．それゆえ，表 1 に掲載のソースファイル数と規模は，実際に配布

されているソースファイル数及び規模より小さいものも含まれていることに注意されたい．

3.2 準備 (1): コメント記述率の分布

まず，各ソフトウェアのソースファイルに対してコメント記述率 (CMR) を測定し，その分布を調べた．紙面の都合上詳細は割愛するが，全体として CMR 値の分布にソフトウェアやバージョンによる大きな差異は見られず，CMR 値の分布は右裾の長いかたちで歪んでいた．また， $CMR = 0$ ，即ち，メソッド定義の内部にコメントが一切書かれていないというソースファイルも多く見られ，全体で 39% 程度含まれていた．実際，大半のソースファイルでは CMR 値は 0 若しくは数パーセント程度 (中央値は 0.044) となっていた (図 1 参照；表示の都合上 $CMR \in [0, 2]$ としているが， $CMR > 2$ となるものも少数存在する)．

3.3 準備 (2): コメントアウト率の分布

CMR と同様に，各ソフトウェアのソースファイルに対してコメントアウト率 (COUTR) を測定し，その分布を調べた．その結果，COUTR 値の分布においてもソフトウェアやバージョンによる大きな差異は見られなかった．そして，全体で 87% 程度のソースファイルでは $COUTR = 0$ ，即ち，コメントアウトは含まれていなかった (図 2 参照；ほとんどが $COUTR = 0$ であるため表示の都合上 $COUTR \in [0, 0.1]$ としているが， $CMR > 0.1$ となるものも少数存在する)．

3.4 準備 (3): コード規模と各メトリクス値の関係

一般に，規模の大きい (長い) ソースコードではフォールトの潜在がより強く疑われる傾向にある¹⁵⁾．それゆえ，今回の調査対象ソフトウェアにおいてもコードの規模によってフォールト潜在の可能性が異なっていることが考えられる．また，コメント記述の傾向もコードの規模によって異なっている可能性も考えられる．本論文の目的は，ソースコードに

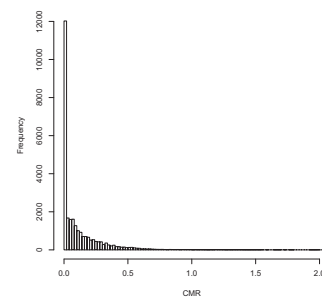


図 1 CMR 値の分布
Fig.1 Distribution of CMR values.

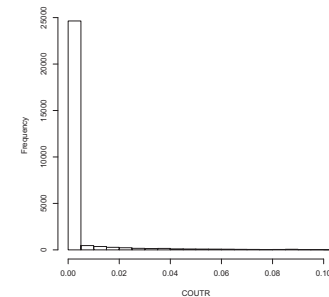


図 2 COUTR 値の分布
Fig.2 Distribution of COUTR values.

4 オープンソースソフトウェアにおけるコメント記述及びコメントアウトとフォールト潜在との関係に関する定量分析

表 2 LOC 値の分布
Table 2 Distribution of LOC values.

最小値	第 1 四分位数	中央値	第 3 四分位数	最大値	最大値 (外れ値 ^{*1} を除く)
2	24	60	146	5228	329

表 3 ソースファイル集合の分割
Table 3 Partition of source file set.

名称	該当するファイル	実際の範囲	ソースファイル数
G_1	最小値 \leq LOC \leq 第 1 四分位数	$2 \leq$ LOC \leq 24	7, 203
G_2	第 1 四分位数 $<$ LOC \leq 中央値	$24 <$ LOC \leq 60	6, 823
G_3	中央値 $<$ LOC \leq 第 3 四分位数	$60 <$ LOC \leq 146	6, 970
G_4	第 3 四分位数 $<$ LOC \leq 最大値 (外れ値を除く)	$146 <$ LOC \leq 329	4, 226

におけるコメント記述とフォールト潜在の関係を定量的に分析することにあるが、そのためにはコード規模が分析に及ぼす影響を無視できない。そこで、コメント記述に関する分析に先立って、コード規模 (LOC) とコメント記述率 (CMR)、コメントアウト率 (COUTR) 並びにフォールト潜在率 (FR) の関係をそれぞれ分析しておく。

表 1 に示した 27,957 個のソースファイルについてコード規模を LOC で測定すると、LOC 値の分布は表 2 の通りであった。ここでは、データ分布 (表 2) に基づいて、ソースファイル集合を表 3 に示す (外れ値を除いた) 四つの部分集合 G_i ($i = 1, 2, 3, 4$) に分割して考える。すると各 G_i におけるコメント記述率 (CMR)、コメントアウト率 (COUTR) 並びにフォールト潜在率 (FR) はそれぞれ表 4 の通りであった。表 4 から分かるように CMR と FR については右肩上がりの傾向、即ち、コード規模が大きくなるにつれてコメント記述率もフォールト潜在率も高くなるという傾向が見られた。 G_i と G_{i+1} における CMR 値の差及び FR 値の差 ($i = 1, 2, 3$) については、いずれの組合せにおいても有意な差であることを有意水準 5% で確認できている。つまり、コードが大きくなるにつれてフォールト潜在の可能性が高まるだけでなく、コメントも多く記述される傾向にあるといえる。なお、COUTR については 3.3 節からも明らかのように、いずれのコード規模でも代表値 (中央値) は 0 であった。

以上のことから、端的にいえば“規模 (LOC) の大きいコードは、コメント記述率 (CMR) もフォールト潜在率 (FR) も高い傾向にある”といえる。したがって、コメント記述の影

*1 第 1 四分位数を Q_1 , 第 3 四分位数を Q_3 とし、 $Q_3 + 1.5(Q_3 - Q_1)$ を超えるものを外れ値¹⁶⁾ とした。

表 4 各ソースファイル集合 G_i における CMR, COUTR 及び FR
Table 4 CMR, COUTR and FR values of G_i .

ファイル集合	CMR (中央値)	COUTR (中央値)	FR
G_1	0	0	0.0780
G_2	0.0313	0	0.1517
G_3	0.0727	0	0.2303
G_4	0.1054	0	0.3495

表 5 コード規模 (G_i) とコメント記述率 (CMR) の各組合せにおけるフォールト潜在率 (FR)
Table 5 FR value in each combination of code-size and CMR categories.

ソースファイル集合	CMR				
	ZERO	有意差	LOW	有意差	HIGH
G_1	0.073	=	0.071	<	0.130
G_2	0.123	=	0.111	<	0.248
G_3	0.162	<	0.198	<	0.312
G_4	0.235	=	0.283	<	0.440

響をより適切に分析するには、コード規模の違いを考慮する必要がある。それゆえ以降では、分析をソースファイル集合 G_i ($i = 1, 2, 3, 4$) ごとに区別して実施し、コード規模の差異による影響を軽減させることにする。

3.5 分析 (1): コメント記述率とフォールト潜在率の関係

ここではコメント記述率 (CMR) とフォールト潜在率 (FR) の関係を分析する。前述したように、ここでの分析はコード規模で分割した $G_1 \sim G_4$ の四つのソースファイル集合に対してそれぞれ実施する。

分析方法は次の通りである：対象ソースファイル集合 G_i をそれぞれ

- CMR = 0 であるもの (便宜上 “ZERO” と表す),
- $\text{CMR} \leq \text{med}[\text{CMR} > 0]$ であるもの (便宜上 “LOW” と表す),
- $\text{CMR} > \text{med}[\text{CMR} > 0]$ であるもの (便宜上 “HIGH” と表す),

という三つに分割して各分割におけるフォールト潜在率 (FR) を算出する。ただし、 $\text{med}[\text{CMR} > 0]$ とは “CMR > 0 であるもの” の中での中央値を意味する。そして、ZERO と LOW の間、並びに LOW と HIGH の間で FR 値に統計的有意差があるかどうかを有意水準 5% で確認する ($i = 1, 2, 3, 4$)。□

分析結果を表 5 に示す。なお、今回の収集データにおいては $\text{med}[\text{CMR} > 0] = 0.126$ であった。表中の “有意差” の列では、統計的有意差を確認できた場合に不等号 (<), 確

5 オープンソースソフトウェアにおけるコメント記述及びコメントアウトとフォールト潜在との関係に関する定量分析

認できなかった場合には等号 (=) をそれぞれ記載している。

表 5 から、フォールト潜在率の分布には“ZERO = LOW < HIGH”あるいは“ZERO < LOW < HIGH”という傾向を確認できた。つまり、コメント記述率の高いソースファイル (HIGH) は、コメントの無いファイル (ZERO) に比べてフォールト潜在率が高い傾向にあるといえる。なお、いずれのコード規模 ($G_1 \sim G_4$) においても、HIGH (CMR > 0.126) でのフォールト潜在率は ZERO (CMR = 0) でのその概ね 2 倍程度であった：

- G_1 : 0.073 → 0.130 (1.78 倍) *1
- G_2 : 0.123 → 0.248 (2.02 倍)
- G_3 : 0.162 → 0.312 (1.93 倍)
- G_4 : 0.235 → 0.440 (1.87 倍)

つまり、コメント記述率が 12.6% を超えるような比較的コメントの多いコードは、コメント記述の無いコードよりもフォールトの潜在が強く疑われるという結果がコード規模を問わず得られた。

3.6 分析 (2): コメントアウト率とフォールト潜在率の関係

3.5 節と同様にして、コメントアウト率 (COUTR) とフォールト潜在率 (FR) の関係を分析した。ここでもソースファイル集合をコード規模に従って $G_1 \sim G_4$ の四つに分割し、それぞれを分析の対象とした。分析方法も 3.5 節と同様であるが、COUTR の場合は大半のコードで COUTR = 0 である (3.3 節参照) こともあり、次のように分析することとした：

対象ソースファイル集合 G_i をそれぞれ

- COUTR = 0 であるもの (便宜上 “ZERO” と表す)、
- COUTR > 0 であるもの (便宜上 “NON-ZERO” と表す)、

という二つに分割して各分割におけるフォールト潜在率 (FR) を算出する。そして、ZERO と NON-ZERO の間で FR 値に統計的有意差があるかどうかを有意水準 5% で確認する ($i = 1, 2, 3, 4$)。 □

分析結果を表 6 に示す。表中の“有意差”の列では、統計的有意差を確認できた場合に不等号 (<) を記載している。

各 G_i において NON-ZERO でのフォールト潜在率 (FR) は ZERO よりも高く、コード規模が同じ程度であってもコメントアウトを含んだコードの方がフォールトの潜在が疑わ

表 6 コード規模 (G_i) とコメントアウト率 (COUTR) の各組合せにおけるフォールト潜在率 (FR)
Table 6 FR value in each combination of code-size and COUTR categories.

ソースファイル 集合	COUTR		
	ZERO	有意差	NON-ZERO
G_1	0.076	<	0.333
G_2	0.143	<	0.326
G_3	0.219	<	0.309
G_4	0.340	<	0.376

れるという傾向を確認できた。ただし、表 6 から分かるように、コード規模が大きいほど、その傾向は弱いといえる。

3.7 分析 (3): コメント記述率とコメントアウト率の組合せに関する分析

最後に、コメント記述率 (CMR) とコメントアウト率 (COUTR) の組合せについて、3.5 節及び 3.6 節と同様にフォールト潜在率 (FR) を算出して比較を行う。前小節までの分析結果として、 $G_1 \sim G_4$ いずれの規模においても次の二つの傾向が共通して確認できている：

- (1) CMR が ZERO のコード (CMR = 0) よりも HIGH のコード (CMR > 0.126) の方がフォールト潜在率 (FR) が高い。
- (2) COUTR が ZERO のコード (COUTR = 0) よりも NON-ZERO のコード (COUTR > 0) の方がフォールト潜在率 (FR) が高い。

そこでこれらを組合せた視点でも分析を行った。

分析結果を表 7 に示す。表 7 ではソースファイルの規模 $G_1 \sim G_4$ ごとに CMR の二つのカテゴリ (ZERO 及び HIGH) と COUTR の二つのカテゴリ (ZERO 及び NON-ZERO) の組合せについて、フォールト潜在率 (FR) を記載している。また、“有意差”と記した行及び列では、それぞれ隣接する FR 値の間に統計的な有意差があるかどうかを有意水準 5% で確認した結果を記載している。不等号 (<) は有意差あり、等号 (=) は有意差無しを表している。なお、 G_1 で COUTR が NON-ZERO (COUTR > 0) の行に関しては、いずれも該当ファイル数が 30 未満と少なかったため、記載並びに有意差の確認を省略している。

表 7 における各行 (横方向) に着目*2すると、いずれもコメント記述の多い (CMR が HIGH) ファイルのフォールト潜在率 (FR 値) の方がコメント記述の無い (CMR が ZERO) ものよりも高いことが分かる。 G_2 の“COUTR が NON-ZERO”の場合に限っては統計的な有意差までは確認できていないが、この傾向を否定する結果ではないと考える。つま

*1 G_1 において “CMR が ZERO の場合の FR 値が 0.073, HIGH の場合の FR 値が 0.130 であり、後者が前者の約 1.78 倍になっている” という意味である。 G_2, G_3 及び G_4 についても同様である。

*2 COUTR を ZERO または NON-ZERO に固定した場合について考えることに相当する。

6 オープンソースソフトウェアにおけるコメント記述及びコメントアウトとフォールト潜在との関係に関する定量分析

表 7 各分類におけるフォールト潜在率 (FR)
Table 7 FR value in each categories about code-size, CMR and COUTR.

ソースファイル 集合	COUTR	CMR		
		ZERO	有意差	HIGH
G_1	ZERO	0.072	<	0.117
	有意差	—	—	—
	NON-ZERO	—	—	—
G_2	ZERO	0.117	<	0.233
	有意差	<	<	<
	NON-ZERO	0.323	=	0.415
G_3	ZERO	0.159	<	0.298
	有意差	=	<	<
	NON-ZERO	0.202	<	0.377
G_4	ZERO	0.248	<	0.434
	有意差	=	=	=
	NON-ZERO	0.129	<	0.452

り、コメントアウトの有無に関わらず、メソッド内にコメント記述の多い (CMR > 0.126) コードの方がフォールトの潜在がより強く疑われると思われる。

一方、同表の各列 (縦方向) に着目^{*1}すると、COUTR が NON-ZERO の場合の FR 値は COUTR が ZERO の場合よりも必ずしも高いという傾向にはなっていない。つまり、コメントアウトの有無に着目することに一定の価値はあると思われるが、コメント記述率に比べるとその影響力は小さいといえる。

なお、“CMR が HIGH” のもの (表 5 参照) と “CMR が HIGH であり、かつ COUTR が NON-ZERO” のもの (表 7 参照) を比較すると (データ数の都合上 G_1 を除外して)

- G_2 : 0.248 → 0.415 (67% 向上)^{*3}
- G_3 : 0.312 → 0.377 (21% 向上)
- G_4 : 0.440 → 0.452 (3% 向上)

となっている。

以上の分析結果から、CMR と COUTR の両方が独立して同程度にフォールト潜在率と関係しているわけではなく、基本的には CMR の影響が大きいといえる。つまり、CMR 値

*1 CMR を ZERO または HIGH に固定した場合について考えることに相当する。

*3 G_2 において “CMR が HIGH” である場合に FR = 0.248 であるのに対し、これを “CMR が HIGH” であってなおかつ “COUTR が NON-ZERO” であるものに絞り込むと FR = 0.415 となり、FR 値 (フォールト潜在の高さ) が 67% 高くなるという意味である。他も同様である。

が大きい場合にはフォールトの潜在がより強く疑われ、そのコードにコメントアウトも併せて存在するようであればフォールト潜在の疑惑が更に強まるといえる。

最後に、各ソフトウェアにおける個別の傾向についてもふれておく。今回の分析対象では Eclipse のファイル数及び規模が他のソフトウェアに比べて特に大きい (表 1 参照) ため、分析結果全体が Eclipse のみの結果に強い影響を受けている可能性も懸念された。そこで補足として、各ソフトウェアについても 3.5 節及び 3.6 節と同様の分析を個別に実施した^{*2} (付録 A.1 参照)。その結果、FR 値に個体差は見られたものの、CMR に関する傾向は全ソフトウェアで概ね共通していた。また、COUTR に関する傾向は、CMR ほど強い傾向ではなかったが、NON-ZERO の場合の FR 値は ZERO の場合と同等かそれ以上という傾向を確認できた。このことから、今回の分析によって得られた CMR 及び COUTR に関する傾向には一定の一般性があるものと思われる。

4. 関連研究

筆者の知る範囲では、コメント文を主たる分析対象とした研究はほとんど見られない。多くの文献では、コメント文をソースコードから除去しないし無視した上でコード解析やメトリクス測定が行われ、データ分析が行われている。あるいはコメント文記述に関するメトリクスも使われるが、他のメトリクスとともに分析データの一部という位置付けになっている。

コメント文を主対象とした数少ないものとしては Fluri ら¹⁷⁾ や Tenny¹⁸⁾ の研究がある。文献 17) ではソフトウェア発展の過程において、コードとともにコメント文も発展 (加筆ないし修正) されていく様子について定量的な分析を行っている。しかしここでは “コメント記述の割合がコード改変とともにどう変化していくのか” という変化の様子や “どういったエンティティ (if 文等) にコメントが付きやすいか” 等の傾向分析が行われており、本論文のようにフォールト潜在との関係分析やコメントアウトに関する分析は行われていない。

文献 18) では、プログラムの可読性という観点から、手続き (procedure) の構成とコメントの有無がそれぞれ可読性の向上に役立つかどうかの検証が行われている。そこでは実験によりコメントの存在が可読性の向上に役立つことが確認されているが、その実験はプログラムにあらかじめフォールトが含まれないことが確認された上で行われており、本論文のようにフォールト潜在との関係性を論じるものではない。また、プログラムの可読性に関する

*2 CMR と COUTR の組合せについてはソースファイル数が極端に少なくなる場合も多く、統計的有意差の確認が難しいため個別の分析は割愛した。

7 オープンソースソフトウェアにおけるコメント記述及びコメントアウトとフォールト潜在との関係に関する定量分析

研究として Buse ら¹⁹⁾ の報告もあり、ここではコメント文を含むさまざまなコード特性について、それらがプログラムの可読性に関する有用なメトリクスとなりうるかどうかを実験によって検討している。結果として、コメント文には可読性をある程度向上させる効果があることが示されているが、もともと可読性の低いコードにコメントを書くことでその可読性の相殺が行われている場合もあると論じられている。本論文はコードの可読性ではなくフォールト潜在の可能性との関係を論じるものであるため文献 19) とも着眼点は異なるが、可読性の低いコードとコメント文の間での効果の相殺という視点は本論文の動機付け(“消臭剤”としての効果; 1 節参照)に通ずるものであり、今後、可読性という視点も視野にいられた分析も検討していく必要があると思われる。

Fault-prone モジュール予測に関して、コードの内容をトークン単位で抽出し、ベイジアンフィルタによる学習を利用して Fault-prone モジュールを検出する研究がある²⁰⁾。平田ら²¹⁾はその手法をコメント文の内容についても適用し、実行可能文と同様にコメント文も Fault-prone モジュール検出に有用であることを示している。コメント文のフォールト潜在に対する影響を分析するという方向性は本論文と同じであるが、その手法はメトリクスに基づいたものではないため基本的なアプローチが本論文とは異なる。ただし、コメント文の内容に踏み込んだ関連研究として、本研究の今後の発展において参考になるものと思われる。

文献 5) 及び 22) は本論文の先行研究であり、それらでは Eclipse に対する CMR 及び COUTR の測定と FR との関係分析を行っている。しかし、文献 5) ではコード規模の影響が考慮されておらず、また、いずれの文献においても分析が Eclipse に限定されており、しかも CMR と COUTR の組合せについては考慮されていなかった。本論文では分析対象を拡充し、コード規模並びに CMR と COUTR の組合せについても分析を行っている。

5. おわりに

本論文は、ソースコードにおけるコメントの記述量とコメントアウトの存在に着目し、それらとフォールト潜在率との関係を定量的に分析した。

七種類のオープンソースソフトウェア(ソースファイル数 27,957, コード規模 3,787,442 LOC)についてデータ収集と統計解析を行った結果、コメントが 12.6% を超える割合で記述されているようなコード(コメント記述率 > 0.126)では、コメントが全く記述されていないコードに比べてフォールト潜在の可能性が 2 倍程度(正確には 1.78 ~ 2.02 倍)高いという傾向を確認できた。また、前者のコードにコメントアウトも含まれているような場合は、その可能性が更に高いということも確認できた。

本来、コメント文はソースコードの理解を助けるものである。つまり、コメント文の存在は決して品質を下げるものではない。しかし、理解性の低い、あるいは複雑性の高いコードは多くのコメントを必要としてしまうことがあり、それゆえコメント記述の多さとフォールト潜在の可能性の高さがつながるという結果になったものと考えられる。Fowler もリファクタリングにおける“不吉な匂い”の一つとしてコメント文の存在を指摘しており、今回の調査結果はその指摘にも符合するものと考えられる。

また、興味深い結果として、コメントアウトの有無がフォールト潜在率の高さと関係していることも確認できた。オープンソース開発ではコードそのものが公開されており、過去のコード変更も全てリポジトリ上で追跡可能である。そのような環境下で、敢えてコメントアウトとして過去のコード断片を現在のコード内に残しているということにも何らかの意味はあるものと考えられる。今回の調査ではコメントアウトの意味にまで踏み込んだ分析はできておらず、筆者による推察の域を出ないが、そのようなコメントアウトの存在は開発における一種の試行錯誤の現れではないかと考える。コメント文やコメントアウトの内容についてもより詳しく分析し、フォールト潜在との関係を解析していくことが今後の課題である。

最後に本論文の主張がコメント文の記述を否定するものではないことを強調しておく。コメントを多く書くことがコードの品質低下を招くのではなく、品質の低いコードにはコメントが多く書かれている場合があるということであり、本論文はその実証を試みたものである。つまり、コード上の問題箇所(分かりにくい部分や複雑な部分)をカバーするために多くのコメントが書かれている場合も少なくないということである。それゆえ、コメントの多さはフォールト潜在の一つの兆候になっており、そのようなコードではコメントの充実よりもむしろ再コーディングや徹底的なコードレビューを検討すべきであるともいえる。コメント無しでも十分に理解できるようなコードを書くことが理想的ではあるが現実的には難しく、多少のコメント記述は必要な場合も多いと思われる。その中においてコメント文の存在を問題視するのが本論文の主旨ではなく、より多くのコメントを必要とするようなコードは品質管理において特に注意を要する存在にもなり得るという警鐘であることに注意されたい。

参 考 文 献

- 1) Vermeulen, A., Ambler, S.W., Bumgardner, G., Metz, E., Misfeldt, T., Shur, J. and Thompson, P.: *The Elements of Java Style*, Cambridge University Press, New York (2000).
- 2) MISRA-C 研究会(編): 組込み開発者におくる MISRA-C:2004 —C 言語利用の高信頼化ガイド—, 日本規格協会, 東京 (2006).

8 オープンソースソフトウェアにおけるコメント記述及びコメントアウトとフォールト潜在との関係に関する定量分析

- 3) Fowler, M.: *Refactoring: Improving The Design of Existing Code*, Addison Wesley Longman, NJ (1999). (児玉公信, 友野晶夫, 平澤章, 梅澤真史 (訳): リファクタリング—プログラミングの体質改善テクニック, ピアソン・エデュケーション, 東京 (2000)).
- 4) Kernighan, B.W. and Pike, R.: *The Practice of Programming*, Addison Wesley, NJ (1999). (福崎俊博 (訳): プログラミング作法, アスキー, 東京 (2000)).
- 5) 阿萬裕久: オープンソースソフトウェアにおけるコメント文記述とフォールト潜在率との関係に関する実証的考察, ソフトウェアエンジニアリング最前線 2010 (松下 誠, 紫合 治, 編), 近代科学社, 東京, pp.97–100 (2010).
- 6) 野中 誠, 水野 修: fault-prone モジュール予測技法の基礎と研究動向, ソフトウェアエンジニアリングシンポジウム 2010 チュートリアル資料, 情報処理学会シンポジウムシリーズ, Vol.2010, No.2, 情報処理学会, 東京 (2010).
- 7) Apache Ant project: Apache Ant, Apache Software Foundation (online), available from <http://ant.apache.org/> (accessed 2011-05-27).
- 8) Apache Logging Services project: Apache log4j, Apache Software Foundation (online), available from <http://logging.apache.org/log4j/> (accessed 2011-05-27).
- 9) Apache Tomcat project: Apache Tomcat, Apache Software Foundation (online), available from <http://tomcat.apache.org/> (accessed 2011-05-27).
- 10) Apache Velocity Project: Apache Velocity Site, Apache Software Foundation (online), available from <http://velocity.apache.org/> (accessed 2011-05-27).
- 11) Apache Xalan project: Apache Xalan Project, Apache Software Foundation (online), available from <http://xalan.apache.org/> (accessed 2011-05-27).
- 12) Eclipse Foundation: Eclipse Foundation open source community website, Eclipse Foundation (online), available from <http://www.eclipse.org/> (accessed 2011-05-27).
- 13) Apache Software Foundation: Apache Xerces Project, Apache Software Foundation (online), available from <http://xerces.apache.org/> (accessed 2011-05-27).
- 14) Boetticher, G., Menzies, T. and Ostrand, T.: PROMISE Repository of empirical software engineering data <http://promisedata.org/> repository, West Virginia University, Department of Computer Science (2007).
- 15) Fenton, N. and Ohlsson, N.: Quantitative Analysis of Faults and Failures in a Complex Software System, *IEEE Trans.Softw.Eng.*, Vol.26, No.8, pp.797–814 (2000).
- 16) Upton, G. and Cook, I.: *A Dictionary of Statistics, Second Edition*, Oxford University Press, Oxford (2008). (白旗慎吾 (監訳): 統計学辞典, 共立出版, 東京 (2010)).
- 17) Fluri, B., Würsch, M., Giger, E. and Gall, H.C.: Analyzing the co-evolution of comments and source code, *Software Quality Journal*, Vol.17, No.4, pp.367–394 (2009).
- 18) Tenny, T.: Program Readability: Procedures Versus Comments, *IEEE Trans. Softw.Eng.*, Vol.14, No.9, pp.1271–1279 (1988).
- 19) Buse, R.P. and Weimer, W.R.: A Metric for Software Readability, *Proc.2008 Int'l Symp.Softw.Testing and Analysis*, pp.121–130 (2008).
- 20) Mizuno, O. and Kikuno, T.: Prediction of Fault-Prone Software Modules Using a Generic Text Discriminator, *IEICE Trans.Information and Systems*, Vol.E91-D, No.4, pp.888–896 (2008).
- 21) 平田幸直, 水野 修: テキスト分類に基づく Fault-prone モジュール検出法におけるコメント行の影響の分析, 情報処理学会研究報告, Vol.2010-SE-170, No.10, pp.1–8 (2010).
- 22) 阿萬裕久: フォールト潜在予測に向けたコメント文記述及びコメントアウトの定量分析, 電子情報通信学会技術研究報告, Vol.110, No.305, KBSE2010-25, pp.13–18 (2010).
- 23) Gosling, J., Joy, B., Steele, G. and Bracha, G.: *Java Language Specification, The 3rd Edition*, Pearson Education, NJ (2006). (村上雅章 (訳): Java 言語仕様 第3版, ピアソン・エデュケーション, 東京 (2006)).
- 24) Geeknet, Inc.: Azureus/Vuze, Geeknet, Inc. (online), available from <http://sourceforge.net/projects/azureus/> (accessed 2011-05-27).

付 録

A.1 各ソフトウェアに対する傾向分析結果

各ソフトウェアにおける CMR 値及び COUTR 値と FR 値との関係を個別に分析した結果を表 8 に示す. 表の見方については表 5 及び表 6 に同じである. ただし, ここでは一定の標本数 (> 30) を確保して統計的な有意差について確認するために, ソースファイル集合 G_2 と G_3 の和集合についてのみ分析を行った. 表中の “—” はデータ数が 30 に満たなかったものを意味する.

表 8 コメント記述率 (CMR) とコメントアウト率 (COUTR) に対するフォールト潜在率 (FR)
Table 8 FR value in each category of CMR and COUTR.

ソフトウェア	CMR			COUTR		
	ZERO	有意差	HIGH	ZERO	有意差	NON-ZERO
Apache Ant	0.106	<	0.205	0.149	=	0.109
Apache log4j	0.475	=	0.538	—	—	—
Apache Tomcat	0.035	=	0.089	0.029	<	0.133
Apache Velocity	0.508	<	0.722	—	—	—
Apache Xalan	0.491	<	0.631	0.545	<	0.607
Eclipse	0.062	<	0.198	0.128	=	0.121
Xerces	0.067	<	0.333	—	—	—