クローンメトリックスを用いた fault-prone モジュール判別の追実験

亀井　靖高[†]（正員）　左藤　裕紀[†]
門田　暁人[†]（正員）　川口　真司[†]
上野　秀剛[††]（正員）　名倉　正剛[†*]
松本　健一[†]（正員）

A Replicated Experiment to Fault-Prone Module Detection with Clone Metrics

Yasutaka KAMEI[†], *Member*, Hiroki SATO[†], *Nonmember*,
Akito MONDEN[†], *Member*,
Shinji KAWAGUCHI[†], *Nonmember*,
Hidetake UWANO[††], *Member*,
Masatake NAGURA[†*], *Nonmember*,
and Ken-ichi MATSUMOTO[†], *Member*

[†] 奈良先端科学技術大学院大学，生駒市
Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan
[††] 奈良工業高等専門学校，大和郡山市
Department of Information Engineering, Nara National College of Technology, Yamatokoriyama-shi, 639–1080 Japan
[*] 現在，（株）日立製作所

あらまし　本論文では，馬場ら [1] によるクローンメトリックスを用いた fault-prone モジュール判別の追実験を行った．Eclipse プロジェクトより収集した 3 バージョン分（バージョン 3.0，3.1，3.2）のモジュールデータを用いた実験の結果，先行研究とは異なり精度の向上は確認できなかった．本論文では，精度が向上しなかった要因を調べるためにクローンメトリックスと fault の関係を分析した．分析結果から，クローンメトリックスは規模の小さいモジュールに対する fault-prone モジュール判別には効果がないものの，馬場らが対象とするようなある程度規模の大きいモジュールに対しては効果があることが示唆された．

キーワード　コードクローン，Fault-prone モジュール，複雑度メトリックス，追実験，ロジスティック回帰

1. まえがき

近年，コードクローン(注1)の研究が盛んに行われており，大規模ソフトウェアからクローンをいかに効率良く検出するか，という段階を経て，検出したクローンを開発にどのように生かすか，という段階にきている．ソフトウェア信頼性向上のための活用事例としては，複雑度メトリックスに加えてクローンの特徴を表す尺度（クローンメトリックス）を fault-prone モジュール判別モデルに用いることで，判別モデルの精度が向上したことが報告されている [1]．報告では，あるプロジェクトで収集された 32 モジュールを判別し，クローンメトリックスを用いない場合と比べて再現率が 0.125，適合率が 0.097 向上したという結果を得ている．

一般的に，fault-prone モジュール判別の精度は，開発ドメインやソフトウェアの規模など開発プロジェクトの特性に左右されることがあるため，実験結果の信頼性向上のために異なるデータセットによる追実験が望ましい．我々の知る限り追実験の報告はなかったため，Eclipse プロジェクトより収集した 3 バージョン分（バージョン 3.0，3.1，3.2）のモジュールデータを用いて評価実験を行ったが，従来研究とは異なり精度の向上は確認できなかった．

本論文では，精度が向上しなかった要因を調べるために，クローンメトリックスと fault の関係を分析した．以降，**2.** で評価実験について詳しく述べ，**3.** で分析結果を説明する．最後に，**4.** で本論文のまとめを述べる．

2. 評価実験

2.1 概　要

馬場ら [1] の追実験として，従来の複雑度メトリックスに加えてクローンメトリックスを説明変数とした fault-prone モジュール判別の精度を評価する．両研究の実験条件を表 1 に示す．本実験では，あるバージョンで発見された fault の履歴をもとに，次のバージョンをリリースする際に fault が含まれるであろうモジュールを判別する．1 ファイルを 1 モジュールとし，モデルの構築には，fault-prone モジュール判別における標準的なモデルの一つであるロジスティック回帰分析を用いた．

本実験で対象としたデータセットは，オープンソースの統合ソフトウェア開発環境の一つである Eclipse プロジェクトのバージョン 3.0，3.1，3.2 を開発中に収集されたメトリックス値を記録したものを用いた．各

表 1　従来研究と本研究の実験条件
Table 1　Evaluation settings.

	馬場らの研究 [1]	本論文
モジュールの粒度	コンポーネント（一つ以上のファイルの集合）	ファイル
モデル構築	ロジスティック回帰分析	ロジスティック回帰分析
モジュール数	40，32	8,313，9,663，11,525
Fault モジュール含有率	約 80%	約 18%
クローンメトリックス	2 種類	5 種類

(注1)：ソースコード中の類似する若しくは同一のコード片の集合 [2].

 ⓒ（社）電子情報通信学会 2010

表 2　計測したメトリックス
Table 2　Measured metrics of Eclipse dataset.

複雑度メトリックス			
名称	概要	名称	概要
TLOC	コードの実行数	NSC	サブクラスの数
MLOC	メソッドの総行数	NSF	静的フィールドの数
NBD	最大ネスト数	NSM	静的メソッドの数
PAR	メソッドのパラメータ数	SIX	特殊化指標の平均
VG	Cyclomatic Complexity	DIT	継承の深さ
NOF	フィールド数	LCOM	凝集性欠落の度合い
NOM	メソッド数	WMC	VG の総和
NORM	オーバライドしたメソッド数		

クローンメトリックス	
名称	概要
NOC	コードクローンとなっているコード片の個数
ROC	全トークンのうちコードクローンとなっているトークンの割合
LEN	当該クローンに含まれるトークンの個数
NIF	当該クローンが含まれているモジュールの数
McC	当該クローンに含まれる条件分岐と繰返し構文の数

表 3　Fault-prone モジュール判別の結果
Table 3　Experiment result.

	加えたクローンメトリックス					判別精度（3.0 → 3.1）		
番号	NOC	ROC	LEN	NIF	McC	再現率	適合率	F1 値
1						.135	.639	.222
2	o	o				.130	.628	.215
3	o					.128	.625	.212
4		o				.131	.636	.218
5			o	o	o	.136*	.570	.219
6			o			.136*	.642*	.224*
7				o		.133	.570	.216
8					o	.136*	.642*	.224*
9	o	o	o	o	o	.131	.561	.213
	加えたクローンメトリックス					判別精度（3.1 → 3.2）		
番号	NOC	ROC	LEN	NIF	McC	再現率	適合率	F1 値
10						.198	.629	.302
11	o	o				.203*	.627	.306
12	o					.200	.626	.303
13		o				.203*	.627	.306
14			o	o	o	.200	.624	.303
15			o			.200	.632*	.304
16				o		.200	.624	.303
17					o	.199	.631	.303
18	o	o	o	o	o	.203*	.626	.307*

*各データセットで最も精度が高い組合せ

バージョンのモジュールと fault の関連付けには Gymothy らの方法 [3] を用いて，一つ以上 fault を含むモジュールを fault モジュールとした．各バージョンのモジュール数は 8,313，9,663，11,525 で，fault モジュール含有率は 17.2%，18.6%，18.4%である．

Fault-prone モジュール判別の評価基準として，再現率，適合率，及び F1 値を用いた．再現率，適合率，及び F1 値は，値域 [0,1] をとり，値が大きいほど判別精度が高いことを表す．

2.2　計測したメトリックス

実験で用いたメトリックスの一覧を表 2 に示す．本実験では，馬場ら [1] が用いた NOC，ROC に加えて，三つの代表的なクローンメトリックス LEN，NIF，McC [4] を用いた．各クローンメトリックスの説明は表 2 を参照されたい．

各モジュールの複雑度メトリックスの計測には Eclipse Metrics plugin [5] を用いて，コードクローンの検出，及びクローンメトリックスの計測には CCFinderX [4] を用いた．クローンの非繰返し度を表すメトリックスのしきい値は 0.5 に設定した．LEN，NIF，McC を計測する際，クローンがモジュールに複数個存在する場合には各クローンメトリックスの最大値を用いた．

2.3　実 験 結 果

実験結果を表 3 に示す．メトリックス列の “o” は判別に用いたメトリックスを指しており，番号 1〜9 がバージョン 3.0 から 3.1 を判別した結果，番号 10〜18 がバージョン 3.1 から 3.2 を判別した結果を示す．番号 1 と 10 が複雑度メトリックスのみを用いた実験結果であり，番号 2〜4，11〜13 が従来研究 [1] で用いられているクローンメトリックスの実験結果，番号 5〜9，14〜18 が本研究で新たに追加したクローンメトリックスの実験結果である．

精度が最も高くなったのは，バージョン 3.0 から 3.1 の判別では LEN を追加した場合（番号 6）及び McC を追加した場合（番号 8），バージョン 3.1 から 3.2 の判別では全メトリックスを用いた場合（番号 18）であった．ただし，F1 値の向上は 3.0 から 3.1 の判別では 0.002，3.1 から 3.2 の判別では 0.005 であった．NOC，ROC のみならず LEN，NIF，McC を併せて用いた場合においても，精度が大きく向上することはなかった．これらの結果から，クローンメトリックスは fault-prone モジュール判別の精度向上に寄与しない可能性が示唆された．

3.　分　　析

3.1　概要と方針

Fault-prone モジュール判別に対してクローンメトリックスの効果が確認できなかった原因を調べるために，各バージョンのクローンメトリックスとバグ密度の関係を分析し，バージョン間で比較する．

fault の有無ではなくバグ密度を用いる理由は，ソースコードの規模はクローンメトリックス及び fault の有無との間に少なからず相関があるので，その要因を取り除くためである．一般に，ソースコードの規模が

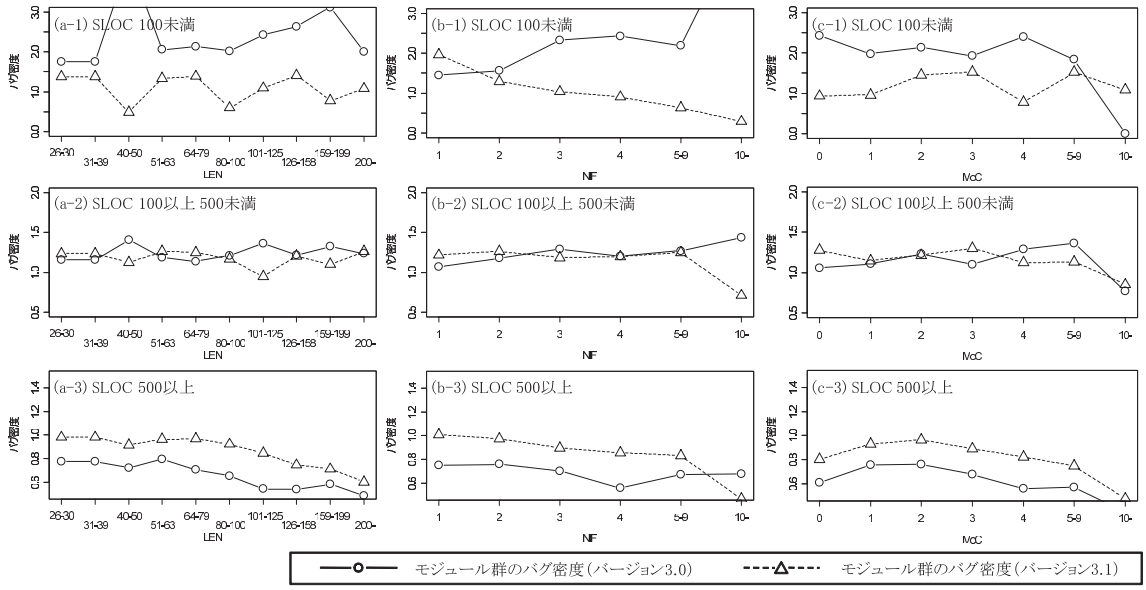

図 1　クローンメトリックスとバグ密度の関係
Fig. 1　Relationship between clone metrics and bug density.

大きくなればなるほど，クローンメトリックスの値は大きくなり，fault が含まれる確率も大きくなる．この場合，クローンメトリックスと fault の有無との間が実際には無関係であっても見かけ上の相関が得られてしまうため，ソースコード規模で正規化した値としてバグ密度を用いた．

ただし，バグ密度を計測し分析する際には次の二つを考慮する．まず，ソースコード規模の大小を考慮する．バグ密度はソースコード規模の大きさによって少なからず影響を受け，ソースコード規模が小さい場合バグ密度は大きく，ソースコード規模が大きい場合バグ密度は小さい傾向がある [6]．そこで，ソースコード規模ごとにモジュールを分類してバグ密度を計測する．

次に，fault モジュール含有率の低さを考慮する．fault を含むモジュールは全体で約 18%であり，多くのモジュールはバグ密度が 0 であるため，モジュール単位でクローンメトリックスとバグ密度の関係を分析することは難しい．そこで，クローンメトリックスをいくつかの階級に分割し，同じ階級に属するモジュールを集めて一つの群とみなし，モジュール群のバグ密度とクローンメトリックスの階級との関係を分析することにした．

3.2　分 析 手 順

手順 1. クローンメトリックスによる分類　クローンメトリックスの大きさごとにクローンを層別し，層別されたクローンを含むか否かでモジュールを分類する．

手順 2. ソースコード規模による分類　手順 (1) で求めたモジュール群を，ソースコード規模によって分類する．

手順 3. バグ密度の計測・比較　手順 (2) で求めたモジュール群ごとにバグ密度を計測・比較する．本論文では，モジュール群に含まれる全 fault 数を，総行数（KSLOC）で割った値をモジュール群のバグ密度（fault 数/KSLOC）とした．

3.3　分析結果と考察

クローンメトリックスによって分類されたモジュール群のバグ密度を図 1 に示す(注2)．横軸はクローンメトリックスの階級を示し，縦軸はバグ密度を示す．一番上の段から順に，SLOC が 100 未満のモジュール，100 以上 500 未満のモジュール，500 以上のモジュールを対象とした分析結果を載せている．

- **バージョン 3.0**　クローンメトリックスとバグ密度の関係は，分析対象のモジュール規模によって異なった．LEN については，(a-1) では，バグ密度は弱い増加傾向（Spearman の順位相関係数が 0.36）であったが，(a-3) では低下傾向（−0.89）であった．NIF については，(b-1) では増加傾向（0.83）であっ

(注2)：紙面の都合上，バージョン 3.0，3.1 の結果，及び，LEN，NIF，McC の結果のみを示す．

たが，(b-3) では低下傾向（−0.71）であった．McC については，(c-2) では相関が見られなかった（0.14）が，(c-3) では低下傾向（−0.71）が見られた．

以上より，クローンメトリックスとバグ密度の間にはすべてのモジュール規模に共通した関係は見られず，fault-prone モジュール判別に対する効果がなかったことを裏づけている．

- **バージョン 3.1**　バージョン 3.0 と同様に，クローンメトリックスとバグ密度の関係は，分析対象のモジュール規模によって異なった．LEN については，(a-1) では相関が見られなかった（−0.17）が，(a-3) では低下傾向（−0.91）が見られた．NIF については，(b-2, b-3) ではどちらも低下傾向を示すものの，相関の強さに大きな違いがあった（−0.49，−1.00）．McC については，(c-1) ではバグ密度は弱い増加傾向（0.29）であったが，(c-3) では低下傾向（−0.61）であった．以上より，バージョン 3.0 と同様に，クローンメトリックスとバグ密度の間にはすべてのモジュール規模に共通した関係は見られなかった．
- **バージョン 3.0 と 3.1 の比較**　LEN，NIF，McC において，SLOC100 未満のモジュールを対象とした場合，各バージョンのバグ密度は推移の傾向が異なった．LEN については，バージョン 3.0 では増加傾向（0.36）を示すものの，バージョン 3.1 では相関が見られなかった（−0.17）．NIF については，バージョン 3.0 では増加傾向（0.83）であったが，バージョン 3.1 では低下傾向（−1.00）であった．McC については，バージョン 3.0 では低下傾向（−0.75）であったが，バージョン 3.1 では弱い増加傾向（0.29）であった．

クローンメトリックスとバグ密度の関係がバージョン間で異なる場合があったため，fault-prone モジュール判別にクローンメトリックスを用いても効果がなかったと考えられる．

一方，SLOC 500 以上のモジュールを対象とした場合，LEN についてはバグ密度は共通して低下傾向を示した(注3)．また，NIF，McC についても，バグ密度の増減はバージョン間で共通する傾向を示した．馬場らの実験ではコンポーネント単位の判別でクローンメトリックスの効果が確認されており，本研究では SLOC500 以上のモジュール群を分析した際にバージョン間で共通の傾向が確認された．これは，クローンメトリックスが規模の小さいモジュールに対する fault-prone モジュール判別には効果がないものの，馬場らが対象とするようなある程度規模の大きいモジュールに対しては効果があることを示唆している．

(注3)：バージョン 3.2 においても同様の傾向が見られた．

4. むすび

本論文では，クローンメトリックスを用いた fault-prone モジュール判別の効果を実験的に評価した．Eclipse プロジェクトから収集したモジュールデータによる追実験の結果，クローンメトリックスは fault-prone モジュール判別の精度向上に寄与しない可能性が示唆された．

クローンメトリックスとバグ密度の関係を分析した結果，(1) クローンメトリックスとバグ密度の間にはすべてのモジュール規模に共通した関係は見られない，(2) クローンメトリックスとバグ密度の関係はバージョン間で異なる傾向を示すことがある，(3) 規模の大きいモジュールでは，クローンメトリックスとバグ密度の関係はバージョン間で共通する傾向を示す，という知見が得られた．(3) は，クローンメトリックスは規模の小さいモジュールに対する fault-prone モジュール判別には効果がないものの，馬場らが対象とするようなある程度規模の大きいモジュールに対しては効果があることを示唆している．

謝辞　本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．また，特別研究員奨励費（課題番号：20009220）の研究助成を受けて行われた．

文　献

[1] 馬場慎太郎, 吉田則裕, 楠本真二, 井上克郎, “Fault-prone モジュール予測へのコードクローン情報の適用,” 信学論（D），vol.91-D, no.10, pp.2559–2561, Oct. 2008.
[2] 肥後芳樹，楠本真二，井上克郎，“コードクローン検出とその関連技術,” 信学論（D），vol.91-D, no.6, pp.1465–1481, June 2008.
[3] T. Gyimothy, R. Ferenc, and I. Siket, “Empirical validation of object-oriented metrics on open source software for fault prediction,” IEEE Trans. Softw. Eng., vol.31, no.10, pp.897–910, 2005.
[4] CCFinderX. http://www.ccfinder.net/index.html
[5] Eclipse Metrics plugin. http://sourceforge.net/projects/metrics
[6] A.G. Koru, D. Zhang, K. El Emam, and H. Liu, “An investigation into the functional form of the size-defect relationship for software modules,” IEEE Trans. Softw. Eng., vol.35, no.2, pp.293–304, 2009.

（平成 21 年 10 月 19 日受付）