Vol.2015-SE-190 No.9
2015/12/15

再現性のあるアソシエーションルールの選定 ～ソフトウェアバグ予測を題材として～

渡部恭史[†1]　門田暁人[†1]　亀井靖高[†2]　森崎修司[†3]

概要：アソシエーションルールマイニングは，事象間の強い関係をデータセットから相関ルールとして抽出する手法であり，得られたルールに再現性があることが求められる．つまり，将来においてもルールに当てはまる事象が頻出することが求められる．本稿では，相関ルールの再現性という概念を整理し，ソフトウェアバグ予測を題材として，再現性の高いルールを抽出する基準を実験的に求めることを目的とする．

キーワード：バグ予測，実験的評価，アソシエーションルールマイニング，データマイニング，ソフトウェア品質

Toward Selection of Reproducible Association Rules ー A Case Study on the Software Defect Prediction

Takashi WATANABE[†1]　Akito MONDEN[†1]
Yasutaka KAMEI[†2]　Shuji MORISAKI[†3]

Abstract: Association rule mining discovers patterns of co-occurrences of attributes as association rules in a data set. The derived association rules are expected to be “reproducible,” i.e. the rules are applicable in future in other data sets. This paper defines the reproducibility of a rule, and aims to find a criteria to distinguish between high reproducible rules and low reproducible ones using a data set for software defect prediction.

Keywords: defect prediction, empirical study, association rule mining, data mining, software quality

1. はじめに

アソシエーションルールマイニング（相関ルール分析）は，事象間の強い関係をデータセットから相関ルールとして抽出する手法であり，従来，POS（Point-Of-Sales）データの分析 1)，Web サイトのアクセスログの分析 21)，タンパク質分析 17)，ソフトウェアバグの分析 11)13)19)など，幅広い分野で用いられている．

得られた相関ルールは，現状の改善や将来の活動に役立てられるが，特に後者の場合，ルールに再現性があることが求められる．つまり，将来においてもルールに当てはまる事象が頻出することが求められる．例えば，ソフトウェアバグに関する相関ルールは，将来のソフトウェア開発におけるバグを予測できることが求められる 11)19)．ところが，再現性の高いルールを選定する方法については，従来ほとんど研究されていない．

本稿では，まず，相関ルールの再現性という概念を整理し，その評価のための「信頼度低下率」という尺度を定義する．そして，再現性に関係する尺度である，ルールの「出現回数」に着目し，どの程度の出現回数であればルールの信頼度低下率を低く抑えることができるかを実験的に明らかにすることを目指す．また，ルールの信頼度低下率は，信頼度にも影響されるため，出現回数，信頼度，信頼度低下率の関係を明らかにすることを目指す．

従来，ルールの支持度，信頼度といった尺度を用いて，ルールを絞り込む方法がよく用いられている．支持度や信頼度は高いルールほど再現性が高いと考えられるが，支持度，信頼度をそれぞれどのような下限値に設定すべきかについて，明確なガイドラインは知られていない．また，後述のように，支持度はデータセットの大きさに依存するため，一般的な下限値を与えることは難しい．本稿では支持度の代替として「出現回数」という尺度を用いる．

相関ルールの抽出および再現性を評価するための題材として，本稿では，ソフトウェアバグ予測，特に，fault-prone モジュール判別のための相関ルール 11)を取り扱う．Fault-prone モジュールとは，バグ（fault）を含む確率の高いモジュール（機能，ソースファイル，サブシステムなど）のことである．過去のソフトウェア開発の実績データから fault-prone モジュールを判別する相関ルールを抽出し，開発中

†1 岡山大学
Okayama University
†2 九州大学
Kyushu University
†3 名古屋大学
Nagoya University

のソフトウェアに適用することで，fault を含みやすい，重点的にテストすべきモジュールの特定に役立てられる 12).

以降，2 章では相関ルール分析について，3 章で Fault-prone モジュール予測における相関ルール分析について 4 章で相関ルールの再現性について，5 章，6 章で実験とその結果，考察について述べる．

2. 相関ルール分析

Agrawal らは頻出する組み合わせ（相関ルール）の抽出方法を文献1)で次のように定義している．販売履歴を対象とした相関ルール抽出の場合，販売履歴をD ，個々の購買をトランザクションT_i，T_i に含まれる1 つの商品をアイテムI_k として，与えられた頻度sよりも大きくDに現れる商品の組合せを$X \Rightarrow Y$ という形式で抽出する．具体的には，$T_i \in D(1 \le i \le n)$，$T_i \subset I$ ，$I = I_1,..., I_k,..., I_m$ (m はユニークなアイテムの数)としたときに， s よりも多い数のT_i を満たす$X \Rightarrow Y$ を求める($X \subset I, Y \subset I, X \wedge Y = \phi$)．ここで， $X \subset T_i \wedge Y \subset T_i$ のとき， T_i は$X \Rightarrow Y$ を満たす．また，X を前提部と呼び，Y を結論部と呼ぶ．

本論文では，各モジュールのソースコードメトリクスやプロセスメトリクスを前提部，fault の有無（fault-prone もしくはnot fault-prone）を結論部として用いる．ただし，ソースコードメトリクスやプロセスメトリクスは量的変数（間隔尺度や比率尺度）である一方，相関ルールで扱える尺度は質的変数（名義尺度や順序尺度）であるため，相関ルール分析を適用する前に各メトリクスを量的変数から質的変数に変換する．例えば，サイクロマティック複雑度の場合，low は[0,10)，medium は[10,30)，high は[30,100)のように3 段階の順序尺度に変換する．すべてのメトリクスが順序尺度に変換された後の相関ルールは「(cyclomatic number=high) and (fan-in= high) ⇒ fault-prone」のように表される．

相関ルール抽出の指標値として以下が知られている．

支持度：対象データにおける相関ルールの出現頻度であり，*support*(X⇒Y)と表記され，*support*(X⇒Y)= s/n である．ただし s=|{T∈D|X⊂T∩Y⊂T} |,n =|{T∈D}|.

信頼度：信頼度は前提部が満たされたときに同時に結論部もみたされる割合であり，*confidence*(X⇒Y)と表記され，*confidence*(X⇒Y) = s/y である．ただし，y =|{T∈D|X⊂T}|.

例えば，X⇒Y というルールにおいて，モジュール数 n=20，条件 X を満たすモジュール数が 10，結論 Y を満たすモジュール数が 8，X と Y を両方満たすモジュール数が 6 の場合，支持度は 0.3（=6/20），信頼度は 0.6（=6/10）となる．

さらに，本論文では，ルールの長さを次の通り定義する．

長さ：ルールの長さは，前提部に含まれる変数の数であり，*length* (X⇒Y)と表記される

3. Fault-prone モジュール予測における相関ルール分析

3.1 相関ルールを用いることの利点

ソフトウェアテストおよび保守において fault-prone モジュール（バグを含む確率の高いモジュール）を特定することは，レビューやテストの効率化やソフトウェアの信頼性を向上するうえで重要である．そのために，モジュールから計測されたメトリクス（プログラム行数，サイクロマティック数，変更行数など）を説明変数とし，モジュールの fault の有無を目的変数とする fault-prone モジュール判別モデルが多数提案されている 9)10)15) ．代表的なモデルとして，線形判別モデル 16)，ロジスティック回帰モデル 17)，分類木 6)，サポートベクターマシン 4)，ランダムフォレスト 8)などが用いられている．

しかし，これらモデルベース手法におけるモデル式は，人間が解釈し理解することが容易でないため，モジュールがバグを含む根拠が分かりにくく，開発現場に受け入れられにくいという問題がある．例えば，最も平易なモデルである線形判別モデルは，下記の式であらわされるが，その解釈は容易でない．

$$y = a_0 + a_1x_1 + a_2x_2 + a_3x_3 + \ldots + a_nx_n$$

ここで，xi は説明変数，ai は判別係数，y は判別値（判別得点）を表し，y が正であれば fault-prone，負のとき not fault-prone であると判別される．

説明変数が互いに独立である場合，判別係数が正ならばその説明変数は fault-prone である確率を高め，負であるときには not fault-prone である確率を高めると解釈できる．しかし，一般に，説明変数は完全には独立ではないため，各変数の係数の正負や大きさだけから解釈を与えることは危険である．このように，最も平易なモデルにおいても，その解釈は容易でなく，サポートベクターマシン，ランダムフォレストのようなより複雑なモデルについては，開発現場の人間が解釈することは極めて困難である．そのため，新しい技術やモデルを開発現場に導入したくない技術者によってしばしば発せられる「この技術（モデル）は，我々のプロジェクトに合うとは思えません.」という言い訳に対し，モデルを採用するよう説得することは容易でない．現実に，これらのモデルを開発現場で採用し，テストの効率化や信頼性向上につなげたという報告はほとんどない．

そこで，ルールベース手法である相関ルール分析を用いた fault-prone モジュール判別が注目されている 11)．相関ルールとして抽出されるルールは，バグを含む（もしくは含まない）条件が解釈しやすい式で表現されるために，開

発現場に受け入れられやすいと期待される．例えば，“(20 ≦ サイクロマティック数) and (10 ≦ ファンイン数) fault prone”というルールは，サイクロマティック数が 20 以上かつ，ファンイン数が 10 以上であるモジュールは fault-prone であることを示しており，直感的にも理解しやすい．

このようなルールのうち，再現性の高いものを過去の開発プロジェクトのデータから抽出することができれば，開発中もしくは将来の開発プロジェクトにおける fault-prone モジュールの予測に役立てることが可能となる．

3.2 相関ルールによる fault-prone モジュール予測

予測を行うにあたって，相関ルールの集合と予測対象のモジュールが与えられたときに，複数のルールが対象モジュールに合致する（つまり，モジュールのメトリクス値が，複数の相関ルールの前提部に合致する）場合があることを考慮しなければならない．合致したルール群の中に，fault-prone と not fault-prone という相反する結論部をもったルールが存在する場合があるためである．このような場合，多数決により結論を決める方法が一般的である 5)．別の方法としては，より長いルールを採用する方法も知られている 19)．予測においては，合致するルールが存在しない場合についても考慮しなければならない．このような場合，モデルによる予測を併用する方法が知られている 5)．

本稿では，上記のような考慮すべき点について，文献 11) の方法に準じて次のように解決する．まず，fault-prone という結論部を持ったルールのみを抽出する（not fault-prone という結論部を持つルールはルール集合から予め除外する）．予測においては，１つ以上のルールに合致した場合 fault-prone と判別し，合致するルールが存在しない場合に not fault-prone と判別する．このように単純化した場合でも，十分な予測精度を得ることができることが示されている 11)．

4. 相関ルールの再現性

4.1 再現性の定義

あるデータセット A から抽出されたルール x の信頼度を Conf(A, x)と表記することとする．本稿では，別のデータセット B におけるルール x の信頼度 Conf(B, x)が Conf(A, x)と同程度であった場合に，ルール x はデータセット B に対して再現性を持つ，と考える．

ただし，再現性は，あり／なしの２値ではなく，Conf(B, x)と Conf(A, x)の差によってその度合いを評価できると考えられる．そこで，Conf(A, x)と比較した Conf(B, x)の低下の度合いを「信頼度低下率」として次の通り定義する．

信頼度低下率= $\frac{Conf(A,x)-Conf(B,x)}{Conf(A,x)}$

信頼度低下率が 0 に近いほど，再現性が高いと判断し，1 に近いほど再現性が低いと判断する．なお，一般に，未知データに対して相関ルールを適用する場合，その判別精度は適合率（precision），再現率（recall）により表現でき，Conf(B,x)は，データセット B にルール x を適用した場合の「適合率」に他ならない．

この定義では，相関ルールの再現性は，ルール単体で決まるのではなく，ルールを適用するデータセット（この場合はデータセット B）に適用して初めて決定されることとなる．このことは，実用上，大きな問題となる．ルール適用前にあらかじめ再現性の高いルールだけを選別したいという要求を満たすことができないためである．

そこで，本稿では，cross-validation によりこの問題を解決する．ルール抽出に用いるデータセット（先の例ではデータセット B）をランダムに２分割し，一方をルール抽出用データセット，もう一方をルールの再現性の評価用データセットとし，信頼度低下率を求める．このようなランダム分割，信頼度低下率の算出を繰り返し行い，信頼度低下率が平均的に大きいルールほど，再現性が低いと判断する．

ただし，cross-validation では，データを分割するたびに毎回異なるルール集合が抽出されるため，個々のルールについて再現性を評価することは容易でない．そこで，本稿では，再現性の高いルールの集合，及び，低いルールの集合について，それぞれの特徴を明らかにすることで，ルールの選別を可能とする（詳しくは次節で述べる）．

4.2 相関ルールの再現性に影響する尺度

4.2.1 ルールの出現回数

相関ルールの特徴量のうち，再現性に大きな影響を与えるものとして，ルールの「支持度」が挙げられる．2 章で述べたように，支持度は，データセット中のルールの出現頻度を表す尺度である．支持度の高い，つまり，頻出する事象を表すルールは，滅多に出現しない事象を表すルールと比べて，再現性が高いと考えられる．

ただし，再現性の高いルールを選別する基準として，支持度を直接用いることは適切ではない．なぜならば，支持度はデータセットの大きさ（サンプルサイズ）に依存するためである．サンプルサイズ 100 であれば，3 件に当てはまるルールであれば支持度 0.03 となるが，サンプルサイズ 10000 であれば，300 件に当てはまらないと支持度 0.03 とはならない．前者はわずか 3 件にしかあてはまらないので偶然起こった事象である可能性が高く，ルールの再現性も低いと考えられるが，後者は 300 件に当てはまるため，全くの偶然の事象を表しているとは考えにくく，他のデータセットでも同様の事象が再現される可能性がより高いと考えられる．つまり，同一の支持度のルールであっても，そ

の再現性は，ルール抽出元のデータセットのサンプルサイズに依存することとなる．

そこで本稿では，支持度の代わりに，ルールの「出現回数」を用いる．出現回数の多いルールほど，(データセットのサンプルサイズに関わらず)再現性が高いと期待される．2 章の相関ルールの定義に基づき，出現回数は次のように定義される．

出現回数：*occurrence* (X⇒Y) と表記され，*occurrence* (X⇒Y)= {T∈D|X⊂T∩Y⊂T}

本稿では，前節で述べた cross-validation を用いた方法により，どの程度の出現回数であれば十分な再現性が得られるかを実験的に明らかにする．

4.2.2 ルールの信頼度

出現回数に加えて信頼度もルールの再現性に大きな影響を与えると考えられる．ルールの再現性を表す尺度である信頼度低下率は，Conf(A, x)と比較しての Conf(B, x)の低下の度合いとして与えられるが，ルールの信頼度が低い，すなわち Conf(A, x)がそもそも小さい場合，Conf(B, x)がさらに低下する余地が少ないため，再現性は見かけ上高くなりやすいと考えられる (なお, 一般に, 信頼度が低いルールは，そもそも価値がない点にも注意されたい)．

そこで，本稿では，ルールの信頼度，出現回数，再現性の 3 者の関係についても実験的に明らかにする．

4.2.3 提案方法の限界および利用方法

本方法で選定される「再現性の高いルール」は，あくまでも同一データセットを用いた cross-validation により得られたものであり，未知データに対して常に高い再現性を示すということは保証されない. 一方, cross-validation により特定された「再現性の低いルール」は，同一データセットにおいても再現性が低いものであるから，未知データに対してはさらに再現性は期待できないと言える. このことから，本方法は，ルールに対して高い再現性を保証するものではなく，再現性が低いルールをフィルタリングする用途で用いるべきである．

5. 実験

5.1 実験対象のソフトウェア

実験には，統合開発環境（IDE）である「Eclipse」におけるタスク管理に特化したプラグインである Mylyn のバージョン 2.0 と 3.0 を用いた．

実験に用いたデータセットの概要を表 1 に示す. Mylyn データセットは，最近の fault-prone モジュール予測でしばしば利用されており 2)7)，ソフトウェアの規模としても大きすぎず小さすぎず，また，fault を含むモジュールと含まないモジュールのバランスが良い（ほぼ半数ずつ存在する）ため，実験対象として選んだ．

表 1　Mylyn データセットの統計量

バージョン	モジュール数（fault 有）	モジュール数（fault 無）	Fault を含むモジュールの割合
2.0	663	599	52.5%
3.0	606	896	40.3%

5.2 Fault の計測

各ソースファイルの fault 数の計測には, SZZ アルゴリズム[18]を用いた．このアルゴリズムは，構成管理ツール（CVS）とバグ管理ツールをリンクすることで，各バグの混入時期と混入したソースファイルを特定できる．

5.3 ルールの抽出

ルール削減対象となるルールセットを得るため，Mylyn を 3-fold cross validation によって分割し取り出した解析元となるデータに相関ルール分析を適用し，結論部が”fault-prone（バグあり）”となるルールを抽出した. ルールの抽出には, NEEDLE 13)を用いた. ルール抽出にあたっては，最小ルール出現回数=5，最小信頼度=0.6 を閾値として与えた. また，ルールの最大長さを 2 に設定した

5.4 実験に用いたソフトウェアメトリクス

実験では，fault の有無を目的変数，ソースコードメトリクス 13 個とプロセスメトリクス 3 個の計 16 個のメトリクスを説明変数として用いた．実験に用いたメトリクスの詳細を表 2 に示す．ソースコードメトリクスの計測には，Understand 20)を用いた．プロダクトメトリクスとしては，Moser らの提案するメトリクス 14)を計測した. 計測にあたっては, Eclipse Foundation によって提供されている CVS リポジトリを用いた．

5.5 3-fold cross validation

提案方法では，データセットを 2 分割して cross validation を行うが，本稿では，3-fold cross validation を用い，繰り返し回数を 30 とした．具体的には，元となるデータをランダムに 3 個のサンプル群に分けて，そのうち 1 つをテストデータとし残り 2 つを解析元となるデータとして分け，30 回繰り返して検証行った．

5.6 評価尺度

本実験では以下の 2 つの評価尺度について調べる．

① 信頼度低下率
② Conf(A, x)ごとの Conf(B, x)（precision）

表 2　実験に用いたメトリクス

種別	名前	定義
プロダクトメトリクス	TLOC	Source Lines of Codes
	FOUT	The number of method calls of a file.
	MLOC	LOC executable
	NBD	Nested block depth
	PAR	Number of parameters
	VG	Cyclomatic complexity
	NOF	The number of attributes
	NOM	The number of methods
	NSF	The number of static attributes
	NSM	The number of static methods
	ACD	The number of anonymous type declarations in a file
	NOI	The number of interfaces in a file.
	NOT	The number of classes in a file
プロセスメトリクス	TPC	The number of revisions of a file
	BFC	The number of times a file was involved in a bug-fix transaction in the 3 months before the release
	PRE	The number of pre-release defects in a file in the 3 months before the release

6. 結果，考察

図 1 は Mylyn2.0，図 2 は Mylyn3.0 におけるルールの出現回数と信頼度低下率との関係を示す．それぞれの縦軸は信頼度低下率，横軸はルール抽出元データセットにおけるルールの出現回数を表す．

まず図 1 の結果について述べる．出現回数が 5 以上 10 未満の場合，信頼度低下率の中央値は約 0.12 であり，出現回数が多くなるほど信頼度低下率は低下するが 0 になることはない．また出現回数が 40 以上の場合，信頼度低下率は最低値周辺になる．

次に図 2 の結果について述べる．出現回数が 5 以上 10 未満の場合，信頼度低下率は約 0.22 であり，出現回数が多くなるほど信頼度低下率は低下するが 0 になることはない．また，図 1 と同様，出現回数が 40 以上の場合，信頼度低下率は最低値周辺になる．ここで図 2 の出現回数が少ない場合の信頼度低下率が同様な場合の図 1 よりも大きくなったのは，Mylyn3.0 における fault を含むモジュールの割合が Mylyn2.0 より小さいことが影響している可能性がある．

図 1，図 2 の結果から両者のデータセットにおいて，出現回数 40 以上のルールを適用した場合，信頼度低下率の

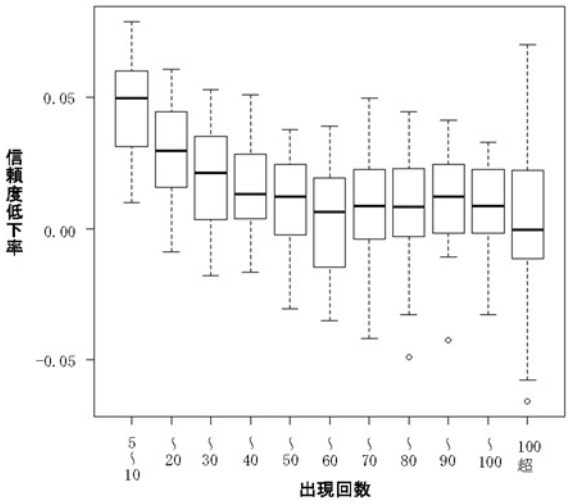

図 1　Mylyn2.0 におけるルールの出現回数と信頼度低下率との関係

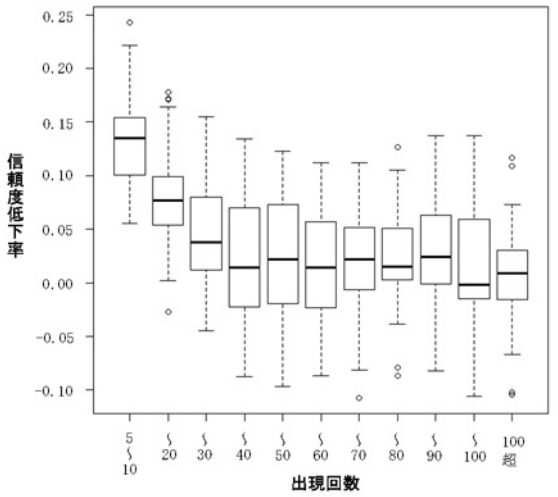

図 2　Mylyn3.0 におけるルールの出現回数と信頼度低下率との関係

最小値周辺を確保できていることが分かる．ただし，4.2.3 節で述べた通り，実際には未知データに対し fault-prone モジュール予測を行うため，出現回数 40 以上のルールを用いたとしても高い再現性を確保できるとは限らない．ただし，出現回数 40 未満のルールであれば，高い再現性は期待できないとは言える．従って，本実験の結果からは，Mylyn のような特徴を持つデータセットの場合，少なくとも出現回数 40 未満のルールでは未知データに対し高い再現度を確保できず，fault-prone モジュール予測を行う上では不要なルールであることが分かった．

次に，ルールの出現回数と信頼度低下率に対する，信頼度の影響を分析する．図 3 は Mylyn2.0，図 4 は Mylyn3.0 の Conf(A, x)ごとのルールの出現回数と Conf(B, x)との関係を示す．両グラフの縦軸は Conf(B, x)の平均値，横軸は出現回

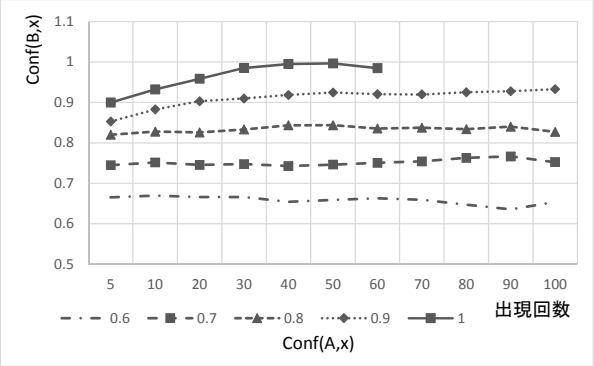

図 3　Mylyn2.0 におけるルールの出現回数，Conf(A,x)，Conf(B,x)の平均値の関係

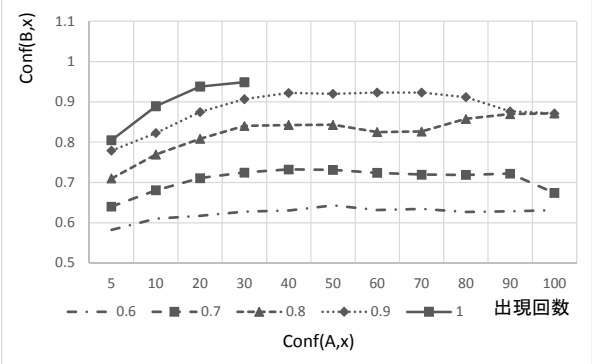

図 4　Mylyn2.0 におけるルールの出現回数，Conf(A,x)，Conf(B,x)の平均値の関係

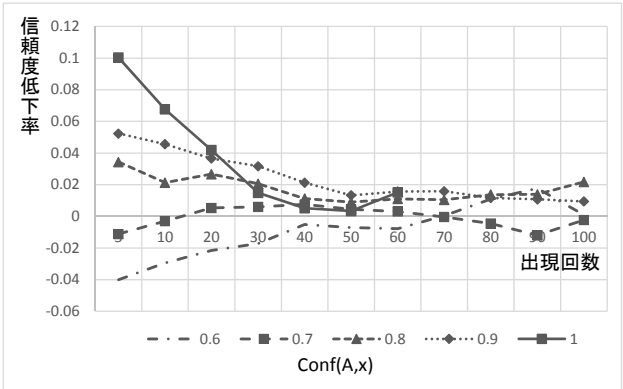

図 5　Mylyn2.0 におけるルールの出現回数，Conf(A,x)，信頼度低下率の平均値の関係

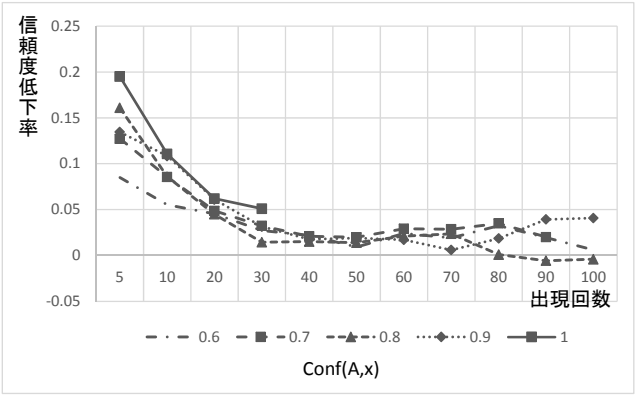

図 6　Mylyn3.0 におけるルールの出現回数，Conf(A,x)，信頼度低下率の平均値の関係

数を表す．また，図 5 と図 6 は，図 4，5 の縦軸を信頼度低下率の平均値に置き換えたものである．

まず図 3 の結果について述べる．Conf(A, x)が 0.6～0.8 の場合，Conf(B, x)は出現回数による変動が小さく，どのような出現回数でも Conf(A,x)に近い，もしくは，わずかに上回る値になっている．また Conf(A, x)が 0.9～1.0 の場合．Conf(B, x)は特に出現回数 5 以上 10 未満の時 Conf(A, x)を大きく下回り，出現回数が多くなるにつれ Conf(A, x)に近づくように増加する傾向にある．このことは，信頼度の低下を抑えるためには，ルールの信頼度が高いほど，より高い出現回数が要求されることを示している．

次に図 4 の結果について述べる．どの Conf(A, x)のグラフに対しても図 1 と比べ Conf(B, x)の変動が見られ，Conf(A, x)が高くなるほど Conf(B, x)の変動は大きくなっている．また Conf(A, x)が 0.7 で出現回数が 100 より多い場合，Conf(A, x)が 0.9 で出現回数が 90 以上 100 未満，100 より多い場合，Conf(A, x)が 1 で出現回数が 50 以上 60 未満の場合 Conf(B, x)が大きく減少している．やはり，信頼度の低下を抑えるためには，ルールの信頼度が高いほど，より高い出現回数が要求される．

図 3 と図 4 でグラフの変動の度合いに違いが表れている理由は Mylyn2.0 と Mylyn3.0 で Fault を含むモジュールの割合が異なることが影響している可能性がある．この点についてのさらなる分析は．今後の課題となる．

なお，グラフが完全には滑らかでない（つまり，ある Conf(A, x)，出現回数で Conf(B, x)が上または下に変動している）理由は，得られたルールの数が少なく外れ値の影響を受けてしまったからだと考えられる．本実験は元となるデータセットをランダムに 2 分割し，一方をルール抽出用データセット，もう一方をルールの再現性の評価用データセットとする作業を 30 回行い得られた結果から平均値を求める．その 30 回において全ての場合で結果が得られる場合は外れ値を含む割合が少なく安定した結果が得られやすい．しかし出現回数が多い，または Conf(A, x)が高い場合は 30 回全てで結果が得られるとは限られないため，外れ値の影響を受けやすいと考える．

次に，図 5 の結果について述べる．Conf(A,x)が 1 で出現回数が 5 以上 10 未満の時，信頼度低下率の値が最も高くなり，Conf(A,x)が高いほど信頼度低下率の値は高くなる傾向があった．ただし，どの Conf(A,x)においても出現回数が

多くなるほど信頼度低下率は 0 周辺に収束する傾向が見られた．

図 6 の結果について述べる．Conf(A,x)が 1 で出現回数が 5 以上 10 未満の時，信頼度低下率の値が最も高くなり，Conf(A,x)が高いほど信頼度低下率の値は高くなる傾向があった．ただしどの Conf(A,x)においても出現回数が多くなるほど信頼度低下率は 0 周辺に収束する傾向が見られた．

図 3 と図 4，図 5 と図 6 の結果から 4.2.2 で前述したように信頼度はルールの再現性に大きな影響を与えることが確認できた．図 3, 図 4 より Conf(A, x)の値が低い場合, Conf(B, x)は Conf(A, x)に近い，もしくはやや上回る値周辺に落ち着き，あまり変動していないことが分かった．また図 5，図 6 より Conf(A,x)の値が大きい場合, 信頼度低下率の値は出現回数が少ない時に大きくなり出現回数が多い時に小さくなることが分かった．これら 4 つのグラフから Conf(A, x)がそもそも小さい場合，Conf(B, x)がさらに低下する余地が少ないため，再現性は見かけ上高くなりやすく，出現回数に影響しにくく，逆に Conf(A, x)の値が大きい場合，Conf(B, x)がまだ低下する余地があるため，再現性は出現回数に影響しやすいということが分かった．

7. おわりに

本論文では，相関ルールによる fault-prone モジュール予測を行う際に，再現性の高いルールを絞り込むための条件について提案した．Mylyn を用いて，提案手法を実験的に評価した．実験から得られた主な知見は以下のとおりである．

- Mylyn のような特徴を持つデータセットの場合，少なくとも出現回数 40 未満のルールは，高い再現度を期待できない．
- Conf(A, x)がそもそも小さい場合，Conf(B, x)がさらに低下する余地が少ないため，再現性は見かけ上高くなりやすい．
- Conf(A, x)の値が大きい場合，Conf(B, x)がまだ低下する余地があるため，再現性は出現回数に影響しやすい．

今後の課題として，未知のデータセットに対しても評価を行うことが挙げられる

謝辞 本研究の一部は，JSPS 科研費基盤研究（C）課題番号 26330086 の補助を受けた．

参考文献

1) Agrawal, R., Imielinski, T. and Swami, A.: Mining Association Rules between Sets of Items in Large Databases, *Proc. Int'l Conf. on Management of Data*, pp.207-216 (1993).
2) D'Ambros, M., Lanza, M., Robbes, R.: An extensive comparison of bug prediction approaches, Proc. 7th IEEE Working Conference on Mining Software Repositories (MSR2010), pp.31-41(2010).
3) Eclipse Mylyn Open Source Project, http://www.eclipse.org/mylyn/
4) Kamei, Y., Monden, A., and Matsumoto, K.: Empirical Evaluation of SVM-based Software Reliability Model, In Proc. 5th ACM-IEEE Int'l Symposium on Empirical Software Engineering (ISESE2006), Vol.2, pp.39-41(2006).
5) Kamei, Y., Monden, A., Morisaki, S., and Matsumoto, K.: A Hybrid Faulty Module Prediction Using Association Rule Mining and Logistic Regression Analysis, *Proc. 2nd Int'l Symposium on Empirical Software Engineering and Measurement*, pp.279-281(2008).
6) Khoshgoftaar, T. M. and Allen, E. B.: Modeling Software Quality with Classification Trees, Recent Advances in Reliability and Quality Engineering, Singapore, World Scientific, pp.247-270 (1999).
7) Lee, T., Nam, J., Han, D., Kim, S., In, H. P.: Micro interaction metrics for defect prediction, Proc. 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering (ESEC/FSE '11), pp.311-321(2011).
8) Lessmann, S., Baesens, B., Mues, C., Pietsch, S.: Benchmarking classification models for software defect prediction: A proposed framework and novel findings, IEEE Trans. on Software Engineering, vol.34, no.4, pp.485-496(2008).
9) Li, P. L., Herbsleb, J., Shaw, M. and Robinson, B.: Experiences and Results from Initiating Field Defect Prediction and Product Test Prioritization Efforts at ABB Inc., Proc. 28th Int'l Conf. on Software Engineering, Shanghai,China, pp.413-422(2006).
10) Mizuno, O., Ikami, S., Nakaichi, S. and Kikuno, T.: Fault-Prone Filtering: Detection of Fault-Prone Modules Using Spam Filtering Technique, Proc. 1st International Symposium on Empirical Software Engineering and Measurement (ESEM'07), pp.374-383(2007).
11) Monden, A., Keung, J., Morisaki, S., Kamei, Y., Matsumoto, K.: A Heuristic Rule Reduction Approach to Software Fault-proneness Prediction, *Proc. 19th Asia-Pacific Software Engineering Conference*, pp.838-847 (2012).
12) Monden, A., Hayashi, T., Shinoda, S., Shirai, K., Yoshida, J., Barker, M., Matsumoto, K.: Assessing the cost effectiveness of fault prediction in acceptance testing, *IEEE Trans. on Software Engineering*, Vo.39, No.1, pp.1345-1357 (2013).
13) Morisaki, S., Monden, A., Matsumura, T., Tamada, H., and Matsumoto, K.: Defect Data Analysis Based on Extended Association Rule Mining, *Proc. 4th Int'l Workshop on Mining Software Repositories*, pp.17-24 (2007).
14) Moser, R., Pedrycz, W., and Succi, G.: A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction," Proc. 30th International Conference on Software engineering (ICSE'08), pp. 181-190 (2008).
15) Munson, J. C. and Khoshgoftaar, T. M.: The Detection of Fault-prone Programs, IEEE Trans. Softw. Eng., Vol.18, No.5, pp.423-433(1992).
16) Ohlsson, N. and Alberg, H.: Predicting Fault-Prone Software Modules in Telephone Switches, IEEE Trans. Softw. Eng., Vol.22, No.12, pp.886-894(1996).
17) She, R., Chen, F., Wang, K., Ester, M., Gardy, J. L., Brinkman, F. L.: Frequent-Subsequence-Based Prediction of Outer Membrane Proteins, Proc. 9th Int'l Conf. on Knowledge Discovery and Data Mining, pp.436-445 (2003).
18) Śliwerski, J, Zimmermann, T., and Zeller, A.: When do changes induce fixes? Proc. Int'l Conference on Mining Software Repositories (MSR'05), pp.1-5(2005).
19) Song, Q., Shepperd, M., Cartwright, M. and Mair, C.: Software Defect Association Mining and Defect Correction Effort Prediction, *IEEE Trans. Softw. Eng.*, Vol.32, No.2, pp. 69-82 (2006).
20) Understand, Scientific Toolworks, Inc., http://www.scitools.com/
21) Yang, Q., Zhangand, H.H. and Li, T.: Mining Web Logs for Prediction Models in WWW Caching and Prefetching, *Proc. 7th Int'l Conf. of Knowledge Discovery and Data Mining*, pp. 473-478 (2001).