NAIST-IS-MT0951090

修士論文

Fault-prone モジュール判別のための 相関ルールの絞り込み

西川 朋希

2012 年 2 月 2 日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
修士（工学）授与の要件として提出した修士論文である.

西川 朋希

審査委員：

松本　健一 教授　　（主指導教員）
関　浩之 教授　　（副指導教員）
門田　暁人 准教授　　（副指導教員）

Fault-prone モジュール判別のための相関ルールの絞り込み*

西川 朋希

内容梗概

ソフトウェア開発において，限られた開発期間で十分な品質を確保することの重要性が高まってきている．近年，モデルベース手法であるロジスティック回帰分析，ランダムフォレスト，サポートベクターマシン等を用いた fault-prone モジュール（バグを含む確率の高いモジュール）判別手法が提案されている．しかし，モデルベース手法は，モデル式を見ても理解しにくいため，開発現場に受け入れられにくい．そこで本論文では，ルールベース手法である相関ルール分析を用いた fault-prone モジュール判別に着目する．相関ルールは「ある条件⇒fault prone」の形式で表され，バグを含む条件が明確である．ただし，ルールが大量に生成されるため，どのルールに着目すればいいのか分からない点が問題である．そこで，本論文では，判別精度をできる限り下げずに相関ルールを絞り込むアルゴリズムを提案する．Eclipse Mylyn を用いた実験の結果，提案アルゴリズムによりルールの数を 6065 個から 62 個に減らすことができ，判別精度を表す F1 値は 0.737 から 0.731 へと 0.006 の低下に抑えることができた．

キーワード

ソフトウェア品質評価，相関ルール分析，判別分析，ソフトウェアメトリクス

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 修士論文，NAIST-IS-MT0951090，2012 年 2 月 2 日

Reduction of Association Rules for Fault-prone Module Detection *

Tomoki Nishikawa

Abstract

While model based software fault predictors, e.g. logistic regression model, Random Forest and Support Vector Machine, are too complex to understand the causes of faults, association rules are much more understandable since rules are described in a simple and intuitive form (condition ⇒ fault prone). Although each association rule is enough comprehensive, usually a huge number of similar rules are often extracted by the association rule mining. This paper proposes a rule reduction algorithm that can eliminate complex (long) and/or similar rules as much as possible without reducing the prediction performance. Through an experiment using the Eclipse Mylyn dataset, the proposed algorithm could reduce the number of rules from 6065 down to 62, while decrease of prediction performance in terms of F1 value was only 0.006 (from 0.737 down to 0.731).

Keywords:

Software quality evaluation, association rule mining, discriminant analysis, software metrics

* Master's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-MT0951050, February 2, 2012.

目次

図目次

表目次

1. はじめに

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[12][14][17]．代表的なモデルとして，線形判別モデル[18], ロジスティック回帰モデル[17], 分類木[9], サポートベクターマシン[7], ランダムフォレスト[11]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，x_i は説明変数，a_i 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，本稿では，ルールベース手法である相関ルール分析[1][20]を用いた fault-prone モジュール判別に着目する．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．このようなルールを過去の開発プロジェクトのデータから抽出することで，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できる．

ただし，相関ルール分析では，ルールが大量に生成される「ルール爆発」の問題が生じる．具体的には，類似するルール群（条件部が類似していたり，包含関係にあるルールの集合）や複雑な（条件部の長い）ルールが大量に生成される．そのため，開発者やテスト実施者にとって，どのルールに着目すればいいのか不明確である点が問題である．従来，支持度，信頼度，リフト値といった指標を用いて，ルールを絞り込む手法がよく用いられているが，この手法ではルールを減らすことはできるが, 類似するルールや複雑なルールが削減されるわけではない．また，ルール削減時に，単純な（条件部の短い）ルールのみを残す手法も考えられるが，短いルールだけでは十分な fault-prone モジュール判別精度が得られる保証がない．

そこで本論文では，次の二つの要求を両立させることを目的とする．

（Req. 1）類似するルールや複雑なルールを減らす．

（Req. 2）予測精度をなるべく落とさない．

これらを両立させるために，本論文では，複雑なルールや類似するルールを削減するアルゴリズムを提案し，その有効性を実験により評価する．提案手法の基本アイデアは, あるルール A の条件がより短いルール B の条件のサブセットであり，かつ，A の信頼度が B の信頼度と比べて十分に高いとはいえない（ある閾値に達しない）場合に，ルール A を削減する．これにより，短くてかつ信頼度の高いルールが優先的に残され, 長さのわりに信頼度が高くないルールが削除される．提案アルゴリズムの有効性を，Eclipse Mylyn データセットを用いて評価する．

以降，2 章で相関ルール分析とその問題点について説明し，3 章でルールを削減するアルゴリズムを提案する．4 章で評価実験の方法について述べ，5 章で実験の結果と考察を述べる．6 章で本稿のまとめと今後の課題を述べる．

2. 相関ルール分析とその問題点

2.1 相関ルール分析

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る．従来，POS(Point-Of-Sales)販売履歴[1]，Webサイトのアクセスログ[24]，タンパク質分析[19]などに用いられてきた．

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よりも多く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以上のトランザクションT_i に満たされる$X \Rightarrow Y$ 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ある．

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 =|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．

例えば，「(cyclomatic number=high) and (fan-in= high) ⇒ fault-prone」というルールは, 前提部に cyclomatic number と fan-in の二つの変数を含むため，その長さは 2 である．

最後に，二つのルール間の関係を次のように定義する．

包含：二つのルール A⇒B と C⇒B について，C⊂A である（C が A を包含する）とき，ルール A⇒B はルール C⇒B を包含する，と呼ぶ．

例えば，「(cyclomatic number=high) ⇒ fault-prone」というルールは，「(cyclomatic number=high) and (fan-in= high) ⇒ fault-prone」を包含する．

2.2 相関ルールによる fault-prone モジュール予測

相関ルール分析によって抽出されたルール群は，しばしば予測に用いられる．一般に，信頼度および支持度が高いルールほど，高い予測性能を持つ．そのため，ルールを抽出する際に，もしくは抽出した後で，信頼度と支持度に閾値（下限値）を設けて，ルールを選定する．本論文では，信頼度と支持度に対する閾値をそれぞれ$\theta_{confidence}$および$\theta_{support}$と表記する．

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[8]．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[20]．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[8].

本論文では，上記のような考慮すべき点について，次のように解決する．まず，ルール集合を小さく保つため，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，5 章で述べるように，本論文では十分な判別精度を得ることができている．

2.3 関連研究

ソフトウェア開発データから相関ルールを抽出し，開発に役立てる研究が行われてきた．Kamei ら[8]は，fault-prone モジュール予測における予測性能向上を目的として，相関ルールとロジスティック回帰分析を併用する方法を提案し，その

有効性を実験により確認している．

Song ら[20]は，開発中に発見・修正された fault の属性データ（fault 混入の原因，fault 修正工数など）から相関ルールを抽出し，同時に発生しやすい fault の種類の予測，および，fault 修正工数のカテゴリ（1 人時以下，1 人時～1 人日，1 人日から 3 人日，3 人日以上）の予測を行った．

Hamano ら[5]は，ソフトウェア開発の混乱（開発コストの超過や，納期の遅延など）の発生につながる条件を，相関ルール分析により明らかにした．

Michail[13]は，ライブラリの再利用のパターンを相関ルール分析により抽出し，得られたパターンを用いてクラスライブラリの作成を試みた．

Morisaki ら[15]は，本来質的尺度にしか適用できない相関ルール分析を拡張し，量的尺度を結論部に持つルールの抽出方法を提案した．拡張相関ルールでは，量的尺度の統計量（平均，標準偏差）および関連する尺度（平均値のリフト値，標準偏差のリフト値）を結論部に持つ．これにより，特徴的な統計量を持つにいたる条件をルールとして抽出できる．

このように，ソフトウェア開発データに対し相関ルール分析を用いて，開発に役立てようという研究は盛んに行われてきた．しかし，次で述べる「ルール爆発」の問題に着目し，ルールの数を効果的に減らそうとする研究は行われていない．

2.4 相関ルール分析の問題点

相関ルール分析の問題は，ルールが大量に生成される「ルール爆発」が生じることである．このことは，次の問題P1，P2，P3を引き起こし，人間によるルールの理解を困難にする．

- (P1) 類似するルールが存在する．例えば，A∧B∧C⇒X とA∧B∧D⇒X のように，条件部の一部だけが異なっているルールが存在する場合，開発者やテスト実施者がどのルールに着目すればいいのかわからなくなってしまう．
- (P2) あるルールが別のルールに包含される．例えば，A∧B⇒XはA∧B∧C⇒Xを包含する．開発者は，この場合条件「∧C」を重視すべきかどうか不明確

となる．

- (P3) 非常に長いルールが存在する．例えば，A∧B∧C∧D∧E∧F⇒X のように条件部が複雑である（長い）場合，開発者による解釈が難しくなり問題である．

従来，信頼度や支持度の高いルールのみを残すことでルールを削減することが広く行われている．本論文でも，支持度について閾値を設けることで，ルールの足切りを行う．一方，信頼度に閾値を設ける方法については，P1～P3の解決に役立たないと考えられる．ここでは，次の４つのルール群について，信頼度の高いルールのみを残す場合について考えてみる．

(Rule #1) $\alpha \cap \beta$ ⇒fault prone 信頼度=82%

(Rule #2) $\alpha \cap \beta \cap \gamma$ ⇒fault prone 信頼度=83%

(Rule #3) $\alpha \cap \beta \cap \varepsilon$ ⇒fault prone 信頼度=92%

(Rule #4) $\lambda \cap \zeta$ ⇒fault prone 信頼度=30%

従来手法では，信頼度の閾値は80%などに設定され，Rule #4が削除される．しかし，Rule #1,2,3は残るため，問題P1,P2,P3のいずれも解決されない．

ルールの長さに閾値を設けることも考えられる．例えば，最大の長さを2とした場合，Rule #2,3が削除される．しかし，ルール#3は高い信頼度を持つため，本来は削除すべきでない．そのため，ルールの長さに閾値を設ける単純な方法は望ましくない．長いルールについては，信頼度が高い場合には採用することを考慮に入れなければならない．

3. 提案手法

3.1 基本アイデア

本論文の目的は，ルール集合が与えられたときに，予測性能をできる限り下げずに，(1)類似するルールを削除する，(2)他のルールに包含されるルールを削除する，(3)長いルールを削除することで，理解しやすい少数のルール集合を得ることである．

基本となるアイデアは，（１）ルールの条件部が長くなることと（２）ルールの信頼度が高くなること，のトレードオフを評価し，長さに見合った信頼度を持たないルールを削除する．例を次に示す．

(Rule #1) α∩β ⇒ fault prone　　信頼度=82%

(Rule #2) α∩β∩γ⇒ fault prone　　信頼度=83%

この例では，ルール#1はルール#2を包含しており，ルール#2は条件∧γの分だけ長くなっているが，信頼度はわずか1%しか向上していない，このような場合，ルール#2は長くて理解しにくいわりに判別精度向上にあまり寄与しないため，残しておく価値は低いため，削除対象とする．次に，別の例を示す．

(Rule #1) α∩β ⇒ fault prone　　信頼度=82%

(Rule #2) α∩β∩ε⇒ fault prone　　信頼度=92%

この例においても，ルール#1はルール#2を包含しており，ルール#2はルール#1に比べて∧εの分だけ長くなっているが，ルール#1に比べて十分に信頼度が高い（10%高い）ため，削除対象としない．本論文では，ルールを削除するか否かを決めるために，ルールが長くなった時の信頼度の増加量について，閾値（下限値）を設定する．この閾値を$\theta_{improvement}$と表記することにする．

次に，包含するルールがない場合について考える．

(Rule #1) α∩β ⇒ fault prone　　信頼度=82%

(Rule #2) γ∩δ⇒ fault prone　　信頼度=82%

(Rule #3) ε∩λ∩ζ⇒ fault prone　　信頼度=80%

この例では, ルール#1とルール#2のいずれもルール#3を包含しない. この場合, ルール#3の信頼度は80%であり, より短いルールであるルール#1とルール#2の信頼度の平均値よりも低いため，削除対象とする.

3.2　ルール削減アルゴリズム

提案するアルゴリズムは，与えられた相関ルール集合に対して，まず，最も短い（たとえば，長さ＝１）ルール群を選択する（つまり，削除対象としない）．次に，２番目に短いルール群（長さ＝２）を調査し，長さ＝１のルールと比べて選択する価値があるかどうかを閾値$\theta_{improvement}$により判定し，削除対象とするルールを決定する. この決定をより長いルール（長さ＝３，４,..）についても順次行い，選択する価値のあるルールがなくなった時点で停止する.

ルール集合 $R=\{R_1 \cup R_2 \cup \cdots \cup R_{\theta_{length}}\}$は，次の閾値を用いた相関ルール分析により得られたものとする.

最小支持度= $\theta_{support}$

最小信頼度= $\theta_{confidence}$

ここで，R_xは，長さ x のルール集合を表す.

このルール集合 R を入力とし，新たな（削減された）ルール集合 R を出力するアルゴリズムを図 1 に示す．この提案アルゴリズムでは，図 1(a)において，長さ x のルール r に対し, ルール x を包含する長さ x-1 のルール群 G(r)を求める. 次に，図 1(b)において，G(r)が空集合のとき，ルール r の信頼度が，長さ x-1 のルールの信頼度の平均値よりも大きいならば r を保持し, そうでないならば r を削除する. G(r)が空集合でないとき, G(r)に含まれるすべてのルールと比較してルール r が十分な信頼度を持つ場合に r を保持し，そうでない場合に r を削除する（図 1(c)）.

Let x = 1 + length of the shortest rule in R

Loop:

For each rule $r \in R_x$ {

Let $G(r) \subset R_{x-1}$ be a set of all rules that comprise r. … (a)

if $G(r) = \emptyset$ {

if confidence(r) < average confidence(R_x) then remove r from R_x. … (b)

}else{

if there exists $g \in G(r)$ that satisfies confidence(r) － confidence(g) < $\theta_{improvement}$ then remove r from R_x. … (c)

}

}

if $R_x = \Phi$ then goto End

Let $x = x + 1$

goto Loop:

End:

図 1　ルール削減アルゴリズム

4. 評価実験

実験では，従来から用いられているルール削減方法（信頼度に閾値を設定することによってルールを削減する）と 3 章で述べた提案手法を用いて，どちらの手法の方がルールを絞り込む手法として適しているのかについて評価する．

4.1 実験対象のソフトウェア

実験には，統合開発環境（IDE）の「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1.0 と2.0 を用いた．バージョン1.0を相関ルール集合の抽出に用い，バージョン2.0 をルール集合による予測性能の評価に用いる．

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[3][10]，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 (リリース日)	Fault を含むモジュール数	Fault を含まないモジュール数	Fault を含むモジュールの割合
1.0 (2006-11-22)	425	598	41.5
2.0 (2007-06-18)	663	599	52.5

4.2 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[22]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[16]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．

表 2 実験に用いたメトリクス

種別	名前	定義
ソースコードメトリクス	TLOC	ソースコード行数
	FOUT	ファンアウトの総数
	MLOC	メソッド行の総数
	NBD	ネストの最大の深さ
	PAR	パラメータの総数
	VG	サイクロマティック複雑度
	NOF	フィールドの総数
	NOM	メソッドの総数
	NSF	静的フィールドの総数
	NSM	静的メソッドの総数
	ACD	匿名のタイプ宣言の総数
	NOI	インターフェースの総数
	NOT	クラスの総数
プロセスメトリクス	TPC	事前変更の総数
	BFC	事前のバグ修正回数
	PRE	リリース前のバグ数

4.3 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[21]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．

4.4 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に，Mylyn のバージョン 1.0 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[15]を用いた．ルール抽出にあたっては，最小支持度$\theta_{support}$=0.01，最小信頼度$\theta_{confidence}$=0.8 を閾値として与えた．また，ルールの最大長さを 5 に設定した．結果として，6065 個のルールが得られた．

表 3 に，抽出したルール集合の特徴を示す．表 3 には長さ 1 のルールは含まれていないが，これは信頼度が 0.8 以上となる長さ 1 のルールが存在しなかったためである．

表 3　削減前のルール集合の特徴

ルールの長さ	ルール数	平均信頼度	平均支持度
1	0	—	—
2	11	0.816	0.141
3	157	0.834	0.108
4	1120	0.844	0.076
5	4777	0.850	0.056

4.5 評価尺度

ルール集合の理解しやすさの尺度として，ルールの数と，ルールの長さの合計（ルール長の総数）を用いた．

また，fault モジュール判別の評価尺度として，再現率，適合率，および F1 値を用いた．再現率は，fault を含む全てのモジュールのうち，fault を含むモジュールと判別した割合を表し，表 4 で示す記号を用いて以下の式で定義される．

$$\text{再現率} = \frac{n_{22}}{n_{21} + n_{22}}$$

また, 適合率は, fault を含むモジュールと判別したモジュールのうち, 実際に fault を含むモジュールである割合を表し，表 4 で示す記号を用いて以下の式で定義される．

$$\text{適合率} = \frac{n_{22}}{n_{12} + n_{22}}$$

fault-prone モジュール判別モデルを評価するためには，再現率と適合率のバランスが重要である. 再現率が高くても, 適合率が低ければ精度が高いとは言えず, 逆に適合率が高くても，再現率が低ければ同じく精度が高いとは言えない．そこで，本実験では，適合率と再現率のバランスを考慮した指標である F1 値を評価基準として用いた．F1 値は，以下の式として定義され，値域は[0,1]となる．F1 値が 1 のとき判別精度が最も高く，0 のとき最も低い．

$$F1 = \frac{2 \times \text{再現率} \times \text{適合率}}{\text{再現率} + \text{適合率}}$$

表 4　判別結果の分類

	Fault を含まないと判別	Fault を含むと判別
実際に fault を含まない	n_{11}	n_{12}
実際に fault を含む	n_{21}	n_{22}

4.6 実験手順

実験については以下に示す 4 つの手順で構成される．

1. **ルールの抽出**　Mylyn のバージョン 1.0 で相関ルール分析を適用し，ルールを抽出する．ルールの抽出には，条件部の変数（16 個）の離散化パラメータは 3，結論部の変数（1 個）の離散化パラメータは 1 に設定．ルールの最大結合数は 4（＝最大長さは 5），最少支持度は 0.01 に設定しておき，支持度が 0.01 以下のルールは足切りしたうえで，ルールを作成する．このとき，結論部が fault-prone であるルールのみを残す．

2. **従来手法の適用**　手順 1．で抽出したルールを最小信頼度を閾値とした絞り込み手法（従来手法）を適用し，ルールを絞り込み，Mylyn のバージョン 2.0 に適用し，判別精度を求める．

3. **提案手法の適用**　手順 1．で抽出したルールを 3 章で提案したルールの絞り込み手法（提案手法）によって，ルールを絞り込み，Mylyn のバージョン 2.0 に適用し，判別精度を求める．

4. **閾値の変化**　それぞれの閾値を変化させて，手順 2．と手順 3．を繰り返し，ルール数（および，ルール長の総数）を減らしていき，それに対応する判別精度を求めていく．

5. 結果と考察

提案手法と従来手法によるルール削減結果を図 2～図 5 に示す．ルール数と F1 値の関係を図 2 に，ルール数と再現率・適合率の関係を図 3 に，ルール長の総数と F1 値の関係を図 4 に，ルール長の総数と再現率・適合率の関係を図 5 に示す．

まず，ルール数に関して，図 2 より，従来手法よりも提案手法の方が，fault-prone モジュールの判別精度（F1 値）を落とさずに，ルール数を削減できた．この原因として，図 3 より，従来手法では，提案手法と比べると，ルール数が減るに従って再現率がより低下していることが挙げられる．一方，適合率に関しては，提案手法，従来手法で大きな差はなかった．

ルール長の総数についても，図 4 より，従来手法よりも提案手法の方が，判別精度（F1 値）を落とさずに，ルール長の総和を削減できた．やはり，提案手法と比べると，従来手法のほうがルール長の総和が減るに従って再現率が低下していたことが原因である（図 5）．

提案手法と従来手法により得られたルール集合の特徴を表 5 と表 6 に示す．表 5 より，提案手法では，閾値$\theta_{improvement}$ = 0.105 の時に 62 個までルールを削減できた（これ以上削減すると，長さ 2 のルールしか残らなくなったため，62 個にまで削減できたことを本論文の成果とする）．このとき，F1 値は 0.731 であり，ルール削減前（6065 個）の場合と比べて，わずか 0.006 の低下に抑えることができた．

一方，表 6 より，従来手法では，閾値$\theta_{confidence}$ = 0.960 のときに 40 個までルールを削減できたが，このときの F1 値は 0.267 であり，予測精度は大幅に低下した．また，提案手法と従来手法によって得られたルールの長さを比較すると，従来手法では，短いルール（長さ 2 と長さ 3）が削除されており，人間にとって解釈しにくい長いルールだけが残されるという結果となった．

以上の結果より，提案手法は従来手法と比べて，ルール数を削減したときに，予測精度の低下を抑えることができ，かつ，理解性の高いルールを残すことができた．

提案手法によって得られた 62 個のルールの詳細を図 6 に示す．これらルールの

解釈方法については本論文の対象外ではあるが，以下では考えられる解釈方法について考察する．

まず，6065 個のルールが 62 個に削減されたものの，人間が理解するには 62 個は依然として多いといえる．個々の変数に着目すると，互いに相関の高い変数が多く含まれており，このことがルールの数を増やす原因となっている．たとえば，TLOC, MLOC, FOUT は互いに相関が高い（相関係数 0.9 以上）．また，TPC, BFC, PRE についても互いに相関が高い（相関係数 0.9 以上）．したがって，これらを同じものとしてルールを解釈するか，あるいは，最初にルール抽出を行う前の段階で，変数を減らしておくことが有効であると考えられる．

ルールの中身に注目すると，大部分のルールにおいて，TPC，BFC，PRE のいずれかが 3（つまり 3 段階中のもっとも値が大きいカテゴリ）であることが条件に含まれる．このことは，過去のバージョンでバグ修正や変更の多いことが，fault-prone モジュールであることの必要条件となっていることを意味する．

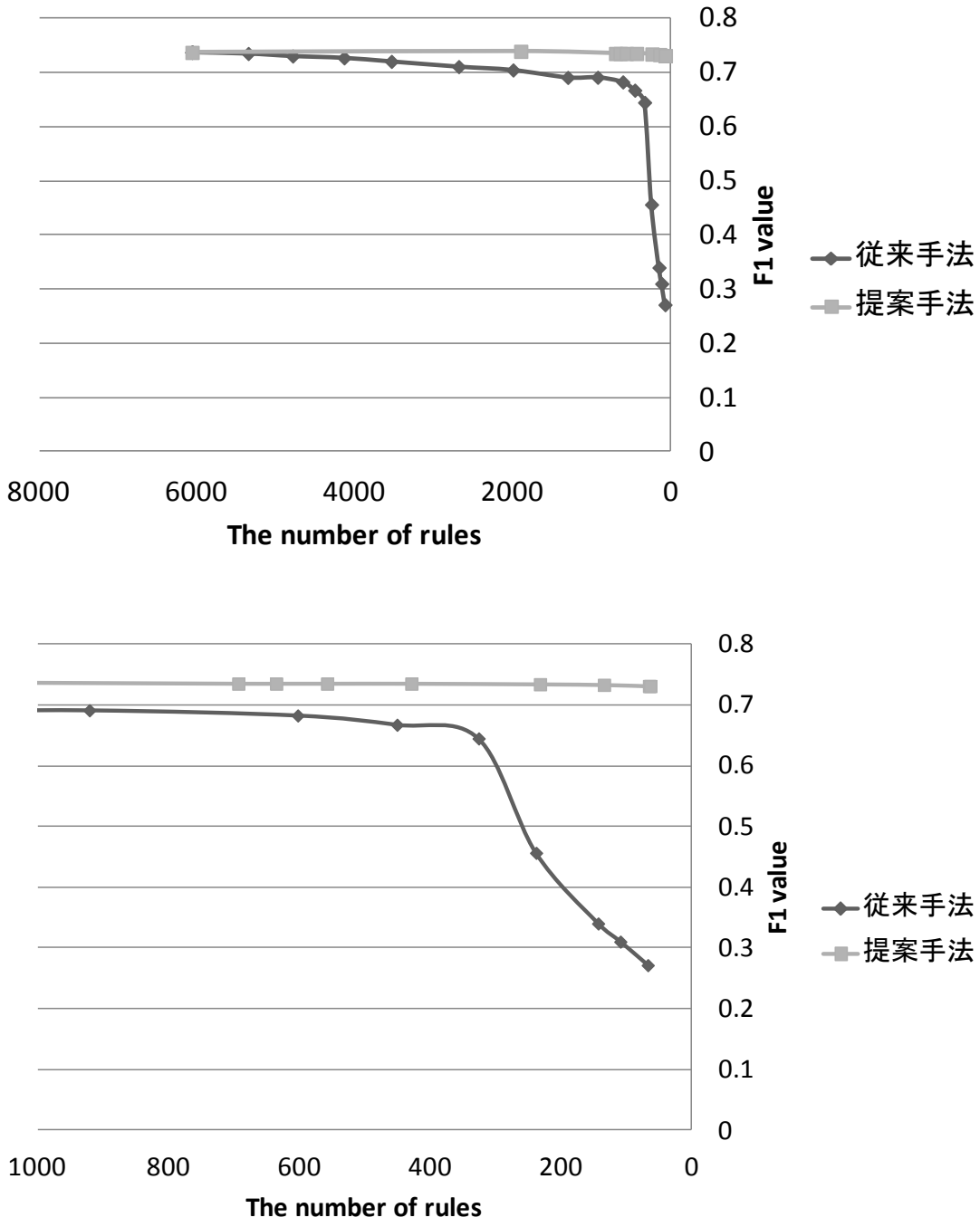

図 2　ルール数と **F1 value** の関係（上）とその拡大図（下）

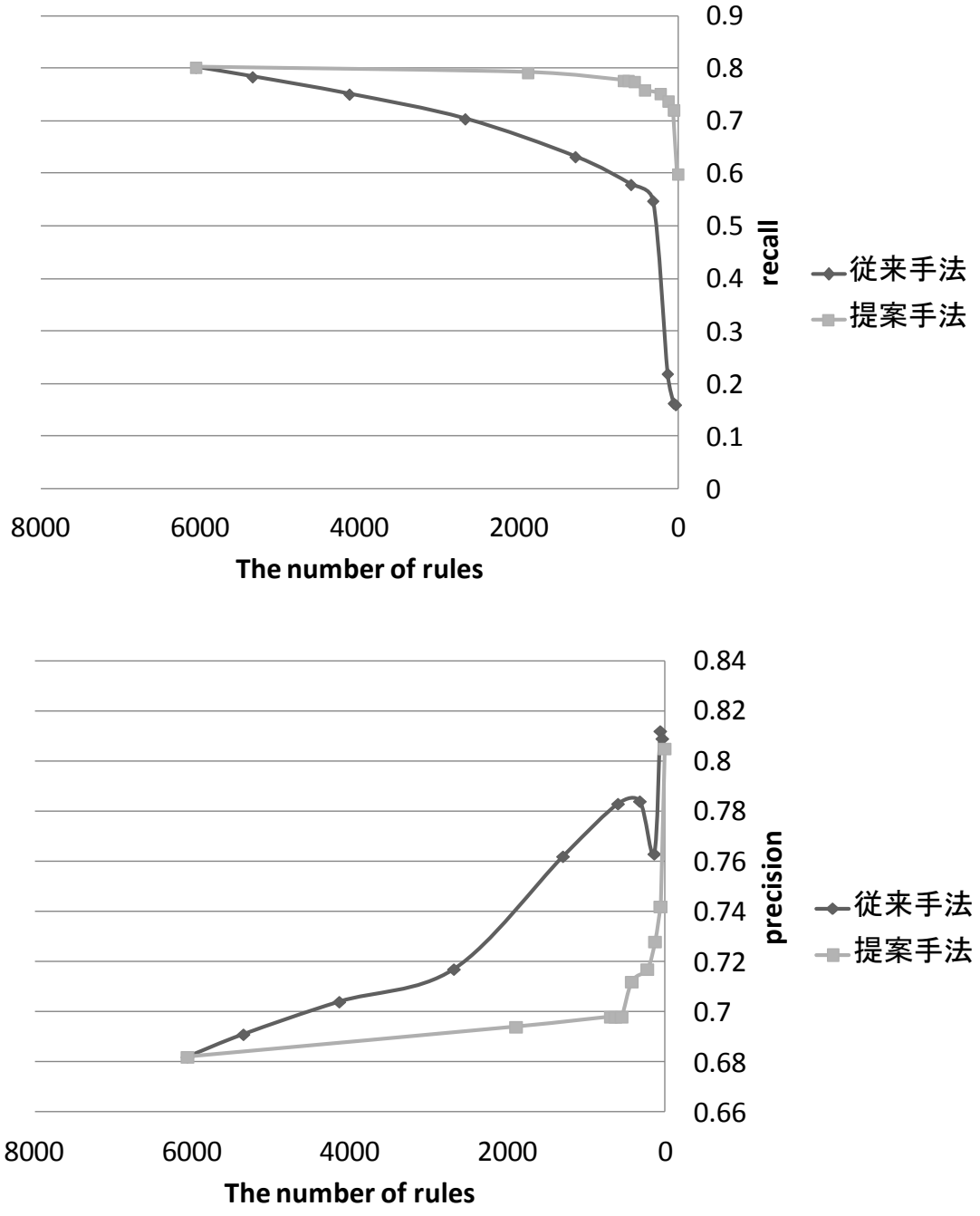

図 3　ルール数と再現率（上）および適合率（下）の関係

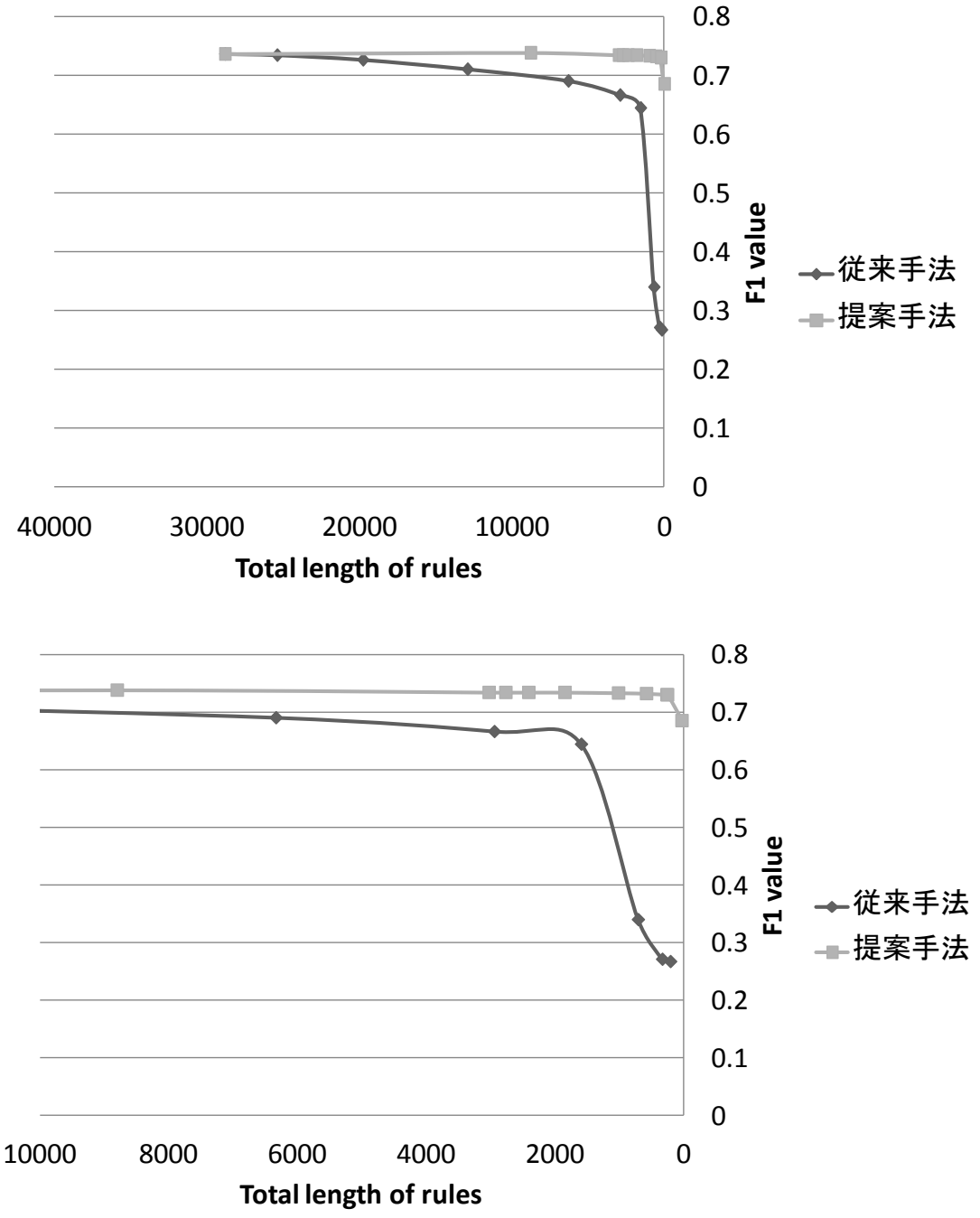

図 **4**　ルール長の総数と **F1 value** の関係（上）とその拡大図（下）

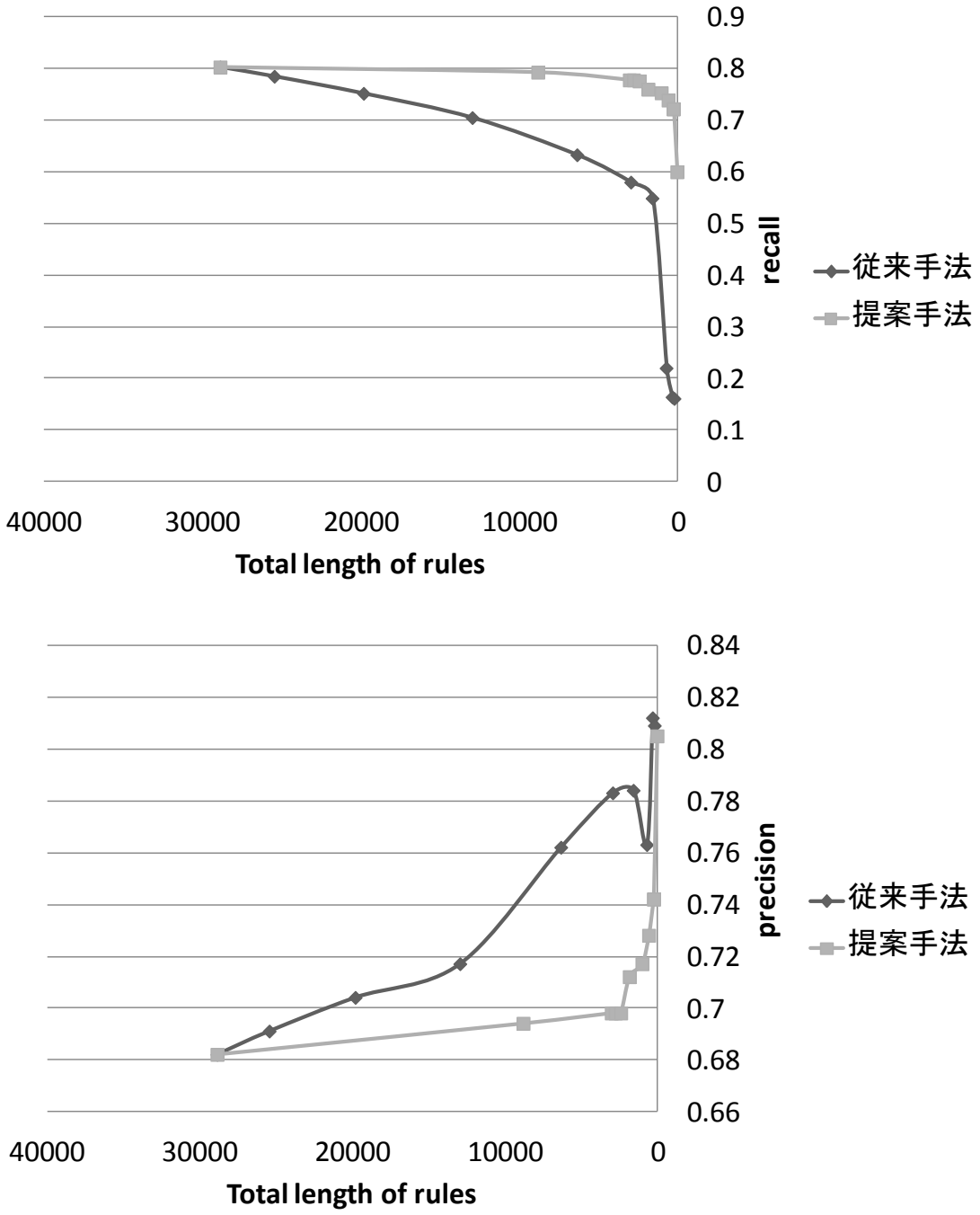

図 5　ルール長の総数と再現率（上）および適合率（下）の関係

表 5　提案手法により得られたルール集合の特徴

閾値 $\theta_{improvement}$	ルールの数					総ルール長	予測精度		
	合計	ルールの長さ					precision	recall	F1
		2	3	4	5				
無し	6065	11	157	1120	4777	28858	0.682	0.802	0.737
0	1896	11	112	423	1350	8800	0.694	0.792	0.739
0.001	693	11	92	228	362	3020	0.698	0.777	0.735
0.003	635	11	88	206	330	2760	0.698	0.777	0.735
0.005	557	11	84	181	281	2403	0.698	0.775	0.735
0.010	428	11	70	126	221	1841	0.712	0.759	0.735
0.030	231	11	30	54	136	1008	0.717	0.752	0.734
0.050	133	11	8	43	71	573	0.728	0.738	0.733
0.100	64	11	1	29	23	256	0.742	0.721	0.731
0.105	62	11	1	29	21	246	0.742	0.721	0.731
0.110	11	11	0	0	0	22	0.805	0.599	0.686

表 6　従来手法により得られたルール集合の特徴

閾値 $\theta_{confidence}$	ルールの数					総ルール長	予測精度		
	合計	ルールの長さ					precision	recall	F1
		2	3	4	5				
無し	6065	11	157	1120	4777	28858	0.682	0.802	0.737
0.810	5354	8	144	1019	4183	25439	0.691	0.784	0.735
0.830	4138	0	76	734	3328	19804	0.704	0.751	0.727
0.850	2684	0	32	413	2239	12943	0.717	0.704	0.711
0.870	1300	0	7	156	1137	6330	0.762	0.632	0.691
0.890	602	0	3	69	530	2935	0.783	0.579	0.667
0.910	325	0	2	37	286	1584	0.784	0.548	0.645
0.930	142	0	0	10	132	700	0.763	0.219	0.340
0.950	66	0	0	6	60	324	0.812	0.163	0.271
0.960	40	0	0	2	38	198	0.809	0.160	0.267

No.	Rule																									
1	VG	=	3.[16	,)	∧	BFC	=	3.[3	,)	⇒	fault prone															
2	TLOC	=	3.[138	,)	∧	TPC	=	3.[3	,)	⇒	fault prone															
3	TLOC	=	3.[138	,)	∧	BFC	=	3.[3	,)	⇒	fault prone															
4	MLOC	=	3.[78	,)	∧	BFC	=	3.[3	,)	⇒	fault prone															
5	MLOC	=	3.[78	,)	∧	TPC	=	3.[3	,)	⇒	fault prone															
6	VG	=	3.[16	,)	∧	TPC	=	3.[3	,)	⇒	fault prone															
7	NOF	=	3.[2	,)	∧	BFC	=	3.[3	,)	⇒	fault prone															
8	NBD	=	3.[2	,)	∧	BFC	=	3.[3	,)	⇒	fault prone															
9	NOF	=	3.[2	,)	∧	TPC	=	3.[3	,)	⇒	fault prone															
10	PAR	=	3.[21	,)	∧	BFC	=	3.[3	,)	⇒	fault prone															
11	NBD	=	3.[2	,)	∧	TPC	=	3.[3	,)	⇒	fault prone															
12	FOUT	=	2.[11	, 33)	∧	MLOC	=	3.[78	,)	∧	TPC	=	3.[3	,)	⇒	fault prone										
13	FOUT	=	2.[11	, 33)	∧	VG	=	2.[6	, 16)	∧	TPC	=	3.[3	,)	∧	PRE	=	2.[0	, 1)	⇒	fault prone					
14	PAR	=	3.[21	,)	∧	VG	=	3.[16	,)	∧	NOM	=	2.[3	, 8)	∧	PRE	=	3.[1	,)	⇒	fault prone					
15	MLOC	=	1.[	, 26)	∧	NBD	=	2.[1	, 2)	∧	VG	=	1.[	, 6)	∧	NSF	=	3.[1	,)	⇒	fault prone					
16	NBD	=	2.[1	, 2)	∧	VG	=	1.[	, 6)	∧	NOM	=	1.[	, 3)	∧	NSF	=	3.[1	,)	⇒	fault prone					
17	NBD	=	2.[1	, 2)	∧	NOM	=	2.[3	, 8)	∧	NSF	=	3.[1	,)	∧	BFC	=	3.[3	,)	⇒	fault prone					
18	NBD	=	2.[1	, 2)	∧	NOM	=	2.[3	, 8)	∧	NSF	=	3.[1	,)	∧	TPC	=	3.[3	,)	⇒	fault prone					
19	VG	=	3.[16	,)	∧	NOM	=	2.[3	, 8)	∧	NSF	=	3.[1	,)	∧	PRE	=	3.[1	,)	⇒	fault prone					
20	FOUT	=	2.[11	, 33)	∧	NOM	=	2.[3	, 8)	∧	NSF	=	3.[1	,)	∧	BFC	=	3.[3	,)	⇒	fault prone					
21	FOUT	=	2.[11	, 33)	∧	NOM	=	2.[3	, 8)	∧	NSF	=	3.[1	,)	∧	TPC	=	3.[3	,)	⇒	fault prone					
22	FOUT	=	3.[33	,)	∧	MLOC	=	2.[26	, 78)	∧	NBD	=	2.[1	, 2)	∧	TPC	=	3.[3	,)	⇒	fault prone					
23	TLOC	=	2.[66	, 138)	∧	NBD	=	2.[1	, 2)	∧	PAR	=	3.[21	,)	∧	VG	=	2.[6	, 16)	⇒	fault prone					
24	FOUT	=	3.[33	,)	∧	MLOC	=	2.[26	, 78)	∧	NBD	=	2.[1	, 2)	∧	BFC	=	3.[3	,)	⇒	fault prone					
25	MLOC	=	2.[26	, 78)	∧	NBD	=	2.[1	, 2)	∧	NOM	=	2.[3	, 8)	∧	BFC	=	3.[3	,)	⇒	fault prone					
26	MLOC	=	2.[26	, 78)	∧	NOF	=	3.[2	,)	∧	NSF	=	3.[1	,)	∧	PRE	=	3.[1	,)	⇒	fault prone					
27	TLOC	=	3.[138	,)	∧	NBD	=	2.[1	, 2)	∧	NSF	=	2.[0	, 1)	∧	BFC	=	2.[1	, 3)	⇒	fault prone					
28	MLOC	=	1.[	, 26)	∧	NBD	=	2.[1	, 2)	∧	NSF	=	3.[1	,)	∧	PRE	=	2.[0	, 1)	⇒	fault prone					
29	TLOC	=	1.[	, 66)	∧	NBD	=	2.[1	, 2)	∧	NOM	=	1.[	, 3)	∧	NSF	=	3.[1	,)	⇒	fault prone					
30	TLOC	=	1.[	, 66)	∧	NBD	=	2.[1	, 2)	∧	VG	=	1.[	, 6)	∧	NSF	=	3.[1	,)	⇒	fault prone					
31	FOUT	=	2.[11	, 33)	∧	PAR	=	2.[7	, 21)	∧	VG	=	3.[16	,)	∧	PRE	=	3.[1	,)	⇒	fault prone					
32	FOUT	=	2.[11	, 33)	∧	VG	=	3.[16	,)	∧	NOF	=	3.[2	,)	∧	PRE	=	3.[1	,)	⇒	fault prone					
33	FOUT	=	3.[33	,)	∧	NBD	=	2.[1	, 2)	∧	NOM	=	2.[3	, 8)	∧	BFC	=	3.[3	,)	⇒	fault prone					
34	FOUT	=	3.[33	,)	∧	NBD	=	2.[1	, 2)	∧	NOM	=	2.[3	, 8)	∧	TPC	=	3.[3	,)	⇒	fault prone					
35	MLOC	=	2.[26	, 78)	∧	NOF	=	3.[2	,)	∧	NOM	=	3.[8	,)	∧	PRE	=	3.[1	,)	⇒	fault prone					
36	NBD	=	3.[2	,)	∧	PAR	=	2.[7	, 21)	∧	NOM	=	3.[8	,)	∧	PRE	=	3.[1	,)	⇒	fault prone					
37	TLOC	=	2.[66	, 138)	∧	FOUT	=	3.[33	,)	∧	NBD	=	2.[1	, 2)	∧	BFC	=	3.[3	,)	⇒	fault prone					
38	TLOC	=	2.[66	, 138)	∧	FOUT	=	3.[33	,)	∧	NBD	=	2.[1	, 2)	∧	TPC	=	3.[3	,)	⇒	fault prone					
39	TLOC	=	3.[138	,)	∧	NBD	=	2.[1	, 2)	∧	NSF	=	2.[0	, 1)	∧	TPC	=	2.[1	, 3)	⇒	fault prone					
40	TLOC	=	1.[	, 66)	∧	NBD	=	2.[1	, 2)	∧	NSF	=	3.[1	,)	∧	PRE	=	2.[0	, 1)	⇒	fault prone					
41	TLOC	=	3.[138	,)	∧	VG	=	2.[6	, 16)	∧	NSF	=	2.[0	, 1)	∧	PRE	=	3.[1	,)	⇒	fault prone					
42	TLOC	=	3.[138	,)	∧	NBD	=	3.[2	,)	∧	NOF	=	2.[0	, 2)	∧	NOM	=	3.[8	,)	∧	BFC	=	3.[3	,)	⇒	fault prone
43	TLOC	=	3.[138	,)	∧	NBD	=	3.[2	,)	∧	NOF	=	2.[0	, 2)	∧	NOM	=	3.[8	,)	∧	TPC	=	3.[3	,)	⇒	fault prone
44	TLOC	=	3.[138	,)	∧	NBD	=	3.[2	,)	∧	NOF	=	2.[0	, 2)	∧	NOM	=	3.[8	,)	∧	PRE	=	3.[1	,)	⇒	fault prone
45	TLOC	=	3.[138	,)	∧	VG	=	3.[16	,)	∧	NOF	=	2.[0	, 2)	∧	NOM	=	3.[8	,)	∧	NSF	=	2.[0	, 1)	⇒	fault prone
46	TLOC	=	3.[138	,)	∧	NOF	=	2.[0	, 2)	∧	NOM	=	3.[8	,)	∧	NSF	=	2.[0	, 1)	∧	PRE	=	3.[1	,)	⇒	fault prone
47	MLOC	=	1.[	, 26)	∧	NBD	=	2.[1	, 2)	∧	PAR	=	2.[7	, 21)	∧	NOF	=	2.[0	, 2)	∧	NSF	=	3.[1	,)	⇒	fault prone
48	MLOC	=	1.[	, 26)	∧	PAR	=	2.[7	, 21)	∧	NOF	=	2.[0	, 2)	∧	NOM	=	1.[	, 3)	∧	NSF	=	3.[1	,)	⇒	fault prone
49	TLOC	=	2.[66	, 138)	∧	NBD	=	2.[1	, 2)	∧	PAR	=	2.[7	, 21)	∧	NOM	=	2.[3	, 8)	∧	BFC	=	3.[3	,)	⇒	fault prone
50	TLOC	=	2.[66	, 138)	∧	NBD	=	2.[1	, 2)	∧	PAR	=	2.[7	, 21)	∧	NOM	=	2.[3	, 8)	∧	TPC	=	3.[3	,)	⇒	fault prone
51	MLOC	=	3.[78	,)	∧	NBD	=	3.[2	,)	∧	NOF	=	2.[0	, 2)	∧	NOM	=	3.[8	,)	∧	NSF	=	2.[0	, 1)	⇒	fault prone
52	MLOC	=	3.[78	,)	∧	NOF	=	2.[0	, 2)	∧	NOM	=	3.[8	,)	∧	NSF	=	2.[0	, 1)	∧	PRE	=	3.[1	,)	⇒	fault prone
53	MLOC	=	3.[78	,)	∧	NBD	=	3.[2	,)	∧	PAR	=	3.[21	,)	∧	NOM	=	2.[3	, 8)	∧	PRE	=	3.[1	,)	⇒	fault prone
54	MLOC	=	3.[78	,)	∧	NBD	=	3.[2	,)	∧	NOM	=	2.[3	, 8)	∧	NSF	=	3.[1	,)	∧	PRE	=	3.[1	,)	⇒	fault prone
55	FOUT	=	1.[	, 11)	∧	NBD	=	2.[1	, 2)	∧	NOF	=	2.[0	, 2)	∧	NSF	=	3.[1	,)	∧	PRE	=	2.[0	, 1)	⇒	fault prone
56	FOUT	=	1.[	, 11)	∧	NBD	=	2.[1	, 2)	∧	NSF	=	3.[1	,)	∧	BFC	=	2.[1	, 3)	∧	PRE	=	2.[0	, 1)	⇒	fault prone
57	FOUT	=	1.[	, 11)	∧	NBD	=	2.[1	, 2)	∧	NSF	=	3.[1	,)	∧	TPC	=	2.[1	, 3)	∧	PRE	=	2.[0	, 1)	⇒	fault prone
58	FOUT	=	3.[33	,)	∧	PAR	=	2.[7	, 21)	∧	VG	=	2.[6	, 16)	∧	NOM	=	2.[3	, 8)	∧	BFC	=	3.[3	,)	⇒	fault prone
59	FOUT	=	3.[33	,)	∧	PAR	=	2.[7	, 21)	∧	VG	=	2.[6	, 16)	∧	NOM	=	2.[3	, 8)	∧	TPC	=	3.[3	,)	⇒	fault prone
60	MLOC	=	3.[78	,)	∧	PAR	=	3.[21	,)	∧	NOM	=	2.[3	, 8)	∧	NSF	=	3.[1	,)	∧	PRE	=	3.[1	,)	⇒	fault prone
61	TLOC	=	1.[	, 66)	∧	FOUT	=	1.[	, 11)	∧	NBD	=	2.[1	, 2)	∧	NSF	=	3.[1	,)	∧	BFC	=	2.[1	, 3)	⇒	fault prone
62	TLOC	=	1.[	, 66)	∧	FOUT	=	1.[	, 11)	∧	NBD	=	2.[1	, 11)	∧	NSF	=	3.[1	, 11)	∧	TPC	=	2.[1	, 11)	⇒	fault prone

図 6　提案手法により得られたルール集合

6. おわりに

本論文では，判別精度をできる限り下げずに相関ルールを絞り込むアルゴリズムを提案した． Eclipse Mylynを用いた実験の結果，得られた主な知見は以下のとおりである．

- ルール数に関して，従来手法よりも提案手法のほうが，fault-proneモジュールの判別精度（F1値）を落とさずに，ルール数を削減できた．詳しくは，従来手法では，$\theta_{confidence}$ = 0.960のときに40個までルールを削減できたが，このときのF1値は0.267であり，予測精度は大幅に低下した．それに対して，提案手法では，$\theta_{improvement}$ = 0.105の時に62個までルールを削減でき，このときのF1値は0.731であり，ルール削減前の場合と比べて，わずか0.006の低下に抑えることができた．

- 再現率に関しては，従来手法に比べると，提案手法の方が，精度の低下が抑えられる結果が得られた．適合率に関しては，F1値，再現率の結果ほどではないが，従来手法と遜色ない結果が得られた．

- ルール長の総数に関しても，F1値，再現率，適合率の推移は，ルール数と同様の結果となった．

- 具体的に削減されたルールは，従来手法では，短く理解しやすいルール（ルール長2と3）が削減されてしまい，理解しにくい長いルール（ルール長4と5）のみが残される結果となった．それに対して，提案手法では，理解性の高い短いルールも残すことができた．以上の結果より，提案手法は従来手法と比べて，ルール数を削減したときに，予測精度の低下を抑えることができ，かつ，理解性の高いルールを残すことができた．

今後の課題として，他のデータセットで同様の予測実験を行うことで，結果の信頼性を向上させることや，提案手法では 62 個まで削減されたものの，人が理解するには依然として多いので，よりわかりやすいルールを構築できる手法を確立することである．

謝辞

本研究を進めるに当たり，多くの方々にご指導，ご協力，ご支援を賜りました．ここに感謝を込めましてお名前を記させていただきます．

奈良先端科学技術大学院大学情報科学研究科 松本健一教授には，論文執筆やプレゼンテーションの作法といった研究に対する直接的な指導に限らず，研究に取り組む上での心構えから，研究者としての在り方など，ありとあらゆる面で熱心なご指導を頂きました．私が本研究を成し遂げることができたのは，先生の御理解と暖かい御助言に励まされたおかげです．先生の御尽力に敬意を表し，心より感謝致します．

奈良先端科学技術大学院大学情報科学研究科 関浩之教授には，本研究を進めるに当たり，貴重なご指導を賜りました．学内発表において多数の御意見，御指摘を頂きました．心より厚く感謝致します．

奈良先端科学技術大学院大学情報科学研究科 門田暁人准教授には，本研究の全過程に対して的確なアドバイスを下さり，研究の進め方，実験の設定についても常に数限り無い適切な御助言と御指導を頂きました．私の研究は，先生の御協力がなければ成し得ませんでした．心より厚く御礼申し上げます．

奈良先端科学技術大学院大学情報科学研究科 大平雅雄助教には，研究者としてあるべき姿について数多くの御指導，御助言を賜りました．研究後期における先生との議論の中で得た炒飯理論は、本研究を完結させる上で欠かせないものとなりました．心より深く感謝致します．

静岡大学情報学部 森崎修司助教には，研究を進めるにあたり，有益なご助言から研究者としての心構えまで，非常に幅広いご指導を頂きました．心より深く感謝致します．

奈良先端科学技術大学院大学情報科学研究科 乾純子秘書には，研究の遂行に必要な事務処理など多岐に渡り手助けいただきました．深く感謝いたします．

九州大学大学院システム情報科学研究院 亀井靖高助教には，本研究を進めるにあたり，熱心な御助力，御指導を頂きました．研究の進め方，実験の設定，論

文執筆やプレゼンテーションに対するアドバイス，研究者としての正しい生活習慣の在り方等，様々な有益な御意見，御指導を頂きました．心より厚く御礼申し上げます．

奈良先端科学技術大学院大学情報科学研究科ソフトウェア工学研究室の皆様には，日頃より多大な御協力と御助言を賜りました．大学院に進学してから，有益な生活が送れたのはひとえに皆様のおかげ様と深く感謝するものでございます．心より厚く御礼申し上げます．

最後に，大学院生活を送るにあたり，私を励まし，経済的にも精神的にもあらゆる面で御支援いただき,温かく見守ってくれた私の家族に心より感謝致します．本当にありがとうございました．

参考文献

[1] Agrawal, R., Imielinski, T. and Swami, A.: Mining Association Rules between Sets of Items in Large Databases, Proc. 1993 ACM SIGMOD Int'l Conf. on Management of Data, pp.207-216 (1993).

[2] Briand, L. C., Basili, V. R. and Hetmanski,C. J.: Developing Interpretable Models with Optimized Set Reduction for Identifying High-Risk Software Components, IEEE Trans. Software Engineering, Vol.19, No.11, pp.1028-1044(1993).

[3] 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).

[4] Eclipse Mylyn Open Source Project, http://eclipse.org/mylyn/

[5] Hamano, Y., Amasaki, S., Mizuno, O. and Kikuno, T.: Application of Association Rule Mining to Analysis of Risk Factors in Software Development Projects, JSSST Computer Software, Vol.24, No.2, pp.79-87(2007).

[6] Herlocker, J. L., Konstan, J. A., Terveen, L. G. and Riedl, J. T.: Evaluating Collaborative Filtering Recommender Systems, ACM Trans. Information Systems, Vol.22, No.1, pp.5-53(2004).

[7] Kamei, Y., Monden, A., and Matsumoto, K.: Empirical Evaluation of SVM-based Software Reliability Model, Proc. 5th ACM-IEEE Int'l Symposium on Empirical Software Engineering (ISESE2006), Vol.2, pp.39-41(2006).

[8] 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 (ESEM2008), pp.279-281(2008).

[9] 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).

[10] 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).

[11] Lessmann, S., Baesens, B., Mues, C., Pietsch, S.: Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings, IEEE Trans. Software Engineering, Vol.34, No.4, pp.485-496(2008).

[12] 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 Conference on Software Engineering, pp.413-422(2006).

[13] Michail, A.: Data Mining Library Reuse Patterns Using Generalized Association Rules, Proc. 22nd Int'l Conference on Software Engineering (ICSE'00), pp.167-176(2000).

[14] Mizuno, O., Ikami, S., Nakaichi, S. and Kikuno, T.: Fault-Prone Filtering: Detection of Fault-Prone Modules Using Spam Filtering Technique, Proc. 1st Int'l Symposium on Empirical Software Engineering and Measurement (ESEM'07), pp.374-383(2007).

[15] 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 (MSR 2007), pp.17-24(2007).

[16] 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 Int'l Conference on Software Engineering (ICSE'08), pp. 181-190 (2008).

[17] Munson, J. C. and Khoshgoftaar, T. M.: The Detection of Fault-prone Programs, IEEE Trans. Software Engineering, Vol.18, No.5, pp.423-433(1992).

[18] Ohlsson, N. and Alberg, H.: Predicting Fault-Prone Software Modules in Telephone Switches, IEEE Trans. Software Engineering, Vol.22, No.12, pp.886-894(1996).

[19] She R., Chen F., Wang K., Ester M., Gardy J.L., Brinkman F.L.: Frequent-Subsequence-Based Prediction of Outer Membrane Proteins, Proc. 9th

ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining, pp. 436-445 (2003).

[20] Song, Q., Shepperd, M., Cartwright, M. and Mair, C.: Software Defect Association Mining and Defect Correction Effort Prediction, IEEE Trans. Software Engineering, Vol.32, No.2, pp. 69-82(2006).

[21] 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).

[22] Understand, Scientific Toolworks, Inc., http://www.scitools.com/

[23] 西川 朋希, 門田 暁人, 森崎 修司, 松本 健一: Fault-Prone モジュール判別のための相関ルールの絞り込み," 情報処理学会研究報告,ソフトウェア工学研究会, Vol.2010-SE-170 , No.20, pp.1 - 6 (2011).

[24] Yang, Q., Zhangand, H.H. and Li, T.: Mining Web Logs for Prediction Models in WWW Caching and Prefetching, Proc. 7th ACM SIGKDD Int'l Conf. of Knowledge Discovery and Data Mining, pp. 473-478 (2001).