情報処理学会論文誌 Vol. 49 No. 12 3954–3966 (Dec. 2008)

相関ルール分析とロジスティック回帰分析を組み合わせたfault-proneモジュール判別方法

亀井靖高[†1] 森崎修司[†1]
門田暁人[†1] 松本健一[†1]

ソフトウェアメトリクスに基づくfault-proneモジュール判別の精度向上を目的として，相関ルール分析とロジスティック回帰分析を組み合わせたfault-proneモジュール判別手法を提案する．提案手法では，与えられたモジュールに対し，重要なルール（支持度，信頼度，または，リフト値の大きなルール）が存在する場合は相関ルール分析によって判別し，そうでない場合は，ロジスティック回帰分析によって判別する．適用可能な複数のルールが存在する場合には，判別結果の多数決を行う．提案手法の判別性能を評価するために，3つの代表的なfault-prone判別モデル（ロジスティック回帰分析，線形判別分析，分類木）の性能と提案手法の性能を比較する実験を行った．実験では，NASA/WVUの公開しているデータセットと，Eclipseプロジェクトから収集したデータセットを対象として，交差検証法による評価と，複数バージョンを用いた評価を行った．実験の結果，重要と見なすルールの選択にはリフト値が適していることが分かり，リフト値に閾値を設けてルールを選定することで，判別精度を表すF1値が従来手法と比較して0.163向上した．

A Faulty Module Detection Method Combining Association Rule Mining and Logistic Regression Analysis

Yasutaka Kamei,[†1] Shuji Morisaki,[†1]
Akito Monden[†1] and Ken-ichi Matsumoto[†1]

To improve the performance of fault-prone module detection, we propose a fault-prone module detection method that combines association rule mining with logistic regression analysis. In our method, if a module satisfies the premise (i.e. the condition in the antecedent part) of one of the important rules (i.e. support, confidence or lift of the rules is large), the module is classified by the rule as either fault-prone or not. Otherwise, the module is classified by the logistic model. We experimentally evaluated the detection performance of the proposed method with different thresholds of each rule interestingness measure (support, confidence and lift) using two module sets (the NASA/WVU dataset and the Eclipse project dataset), and compared it with three well-known fault-proneness models (logistic regression model, linear discriminant model and classification tree). The result showed that the lift was the most suitable measure to select useful association rules in the proposed method compared to other measures (support and confidence). The improvement of the F1-value of the proposed method with the lift was 0.163 at maximum compared to conventional models.

1. はじめに

ソフトウェアテストおよび保守工程において，fault-proneモジュール（faultを含む確率の高いモジュール）[16]を特定しテスト工数を重点的に割り当てることで，テストの効率化，および，信頼性の向上が見込まれる[8),11)]．そのために，モジュールから計測されたメトリクス（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[2),3),7),13),16),19)]．ロジスティック回帰分析，線形判別分析，分類木などのモデル化手法が提案されており，ロジスティック回帰分析が特に広く用いられている[2),3),13)]．

ロジスティック回帰分析では，与えられた説明変数の値の組（モジュールから計測されるメトリクス値の組）に対する，ある現象の発生する条件付き確率（モジュールがfaultを含む確率）をロジスティック回帰式としてモデル化する．一般に，faultの混入は確率的に生じるため，線形判別分析のようにfaultの有無を二者択一で判定するよりも，確率事象としてとらえる方がより妥当であるとの考えから，ロジスティック回帰分析は広く用いられてきた[2),3),13)]．

一方，非モデルベースの方法として，相関ルール分析を用いたfault-proneモジュール判別方法が近年提案されている[19)]．相関ルール分析は，与えられたデータセットから「ある事象Xが発生した場合に高確率で別の事象Y（今回の場合はfaultの含有）が発生する」という法則を相関ルールとして抽出する手法である．たとえば，相関ルールは「$(4 \leq \text{cyclomatic number} < 5)$ and $(3 \leq \text{fan-in} < 5) \Rightarrow$ fault」のように表す．このルールでは，モジュールのサイクロマティック数が4以上で5より小さく，かつ，fan-inが3以上で5より小さいモジュールのfaultの有無を判別することができる．

†1 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

相関ルール分析の特長の 1 つは，fault の混入という事象を（1 つのモデル式ではなく）多数のルールの集合としてとらえるため，多様な fault 混入の要因を抽出し，判別に用いるのに適していることである．もう 1 つの特長としては，各ルールの重要性を表す指標値（支持度，信頼度など）を利用することで，判別精度の向上に寄与するであろうルールのみを選択することができる（3 章）．ただし，選定されたルール集合は，あらゆるモジュールを判定できるとは限らない．モジュールによっては条件の一致するルールが存在せず，fault の有無を判別できない場合がある．

本論文は，ルールベースとモデルベースの判別手法を組み合わせた fault-prone モジュール判別手法の 1 つとして，相関ルール分析とロジスティック回帰分析を組み合わせた fault-prone モジュール判別手法を提案する[⋆1]．ロジスティック回帰分析をモデルベースの手法として用いる理由は，fault-prone モジュール判別における標準的なモデルであるといわれている[2]ため，および，一般に fault の混入は確率的に生じることから，線形判別分析や分類木のように fault の有無を二者択一で判定するよりも，確率事象としてとらえる方がより自然であると考えたためである．提案手法では，与えられたモジュールに対し，重要なルール（支持度，信頼度，または，リフト値の大きなルール）が存在する場合は相関ルール分析によって判別し，そうでない場合は，ロジスティック回帰分析によって判別する．適用可能な複数のルールが存在する場合には，判別結果の多数決を行う．我々の知る限り，ロジスティック回帰分析と相関ルール分析を組み合わせた fault-prone モジュール判別手法は存在せず，また，いずれの指標値が本手法に最も適しているかも明らかではない．さらに，各指標の閾値をどの程度の大きさに設定し，どの程度ルールの絞り込みを行うべきかも明らかではない．

そこで本論文では，いずれの指標値の閾値をどの程度の大きさに設定するべきかを明らかにするために，各指標値の閾値を変化させて，それぞれの判別精度を比較する実験を行う．また，その判別精度を，3 つの代表的な fault-prone 判別モデル（ロジスティック回帰分析，線形判別分析，分類木）と比較対照することで評価する．評価実験では，NASA/WVU が公開しているデータセット（モジュールのメトリクス値と fault の有無が記されたもの）と，Eclipse プロジェクトから収集したデータセットを対象として，交差検証法による評価と，複数バージョン間における評価を行う．

交差検証法は，1 つのデータセットをモデル構築用のフィットデータとモデル評価用のテストデータに分割しモデルの判別精度を評価する方法であり，fault-prone モジュール判別モデルの精度評価に広く行われている[9),17)]．本実験では，NASA（アメリカ航空宇宙局）が公開しているデータセットのうち，評価実験に広く用いられている[9),17)]KC1 プロジェクトのデータセットを用いた（5 章）．

一方，複数バージョン間における評価では，fault-prone モジュールを利用する状況により近い条件での実験を行う．実験では，過去のバージョンで報告された fault を記録したフィットデータを用い，次期バージョンに含まれていた fault の予測を試みた．ただし，NASA が公開しているデータセットには fault が報告された時期が示されていない．そこで我々は，Eclipse プロジェクトからモジュールのメトリクス値と fault の報告の履歴を収集し，Gyimothy らの示す方法[5)] によって 2 バージョン分のデータセットを作成し，実験に用いた（6 章）．

以降，2 章で実験に用いる fault-prone モジュール判別モデルについて説明し，3 章で相関ルール分析について説明する．4 章で相関ルール分析とロジスティック回帰分析を組み合わせた fault-prone モジュール判別手法を提案する．5 章では交差検証法を用いた評価実験，6 章では複数バージョンを用いた評価実験について述べる．7 章で関連研究について述べ，最後に 8 章で本論文のまとめと今後の課題を述べる．

2. Fault-prone モジュール判別モデル

Fault-prone モジュール判別では，過去に開発されたモジュールのメトリクス値と fault の有無から判別モデルを構築する．構築したモデルに対して，モジュールから計測したメトリクス値を入力することで，当該モジュールが fault を含むかどうか判別される．判別モデルの種類は多数あるが，fault-prone モジュール判別においては，下記の 3 つのモデルがよく用いられており，本論文の評価実験においても，比較対象として採用する．

2.1 ロジスティック回帰分析

ロジスティック回帰分析では，判別関数はロジスティック関数の形で表現される．各ケースに対して p 個の説明変数の値が観測されているとき，判別モデル（判別関数）は式 (1) の形となる．

$$P(y|x_1,\cdots,x_p)=\frac{1}{1+e^{-(\alpha_1 x_1+\cdots+\alpha_p x_p+\beta)}} \tag{1}$$

ここで，$y \in \{0,1\}$ は群を表す目的変数，x_i は説明変数，α_i は判別係数であり，$P(y|x_1,\cdots,x_p)$ は，説明変数の値の組 $x_1,\cdots,x_p$ に対して，y が 1 の値をとる条件付き確率である．本論文では $P(y|x_1,\cdots,x_p)$ が 0.5 以上の値をとる場合，fault-prone モジュー

⋆1 本論文は，筆者らのシンポジウム原稿[24)] に実験を追加し，論文としてまとめたものである．

ルであると判定する．

2.2　線形判別分析

線形判別分析では，判別関数はケースの特性を表す説明変数の線形結合により表される．各ケースに対して p 個の説明変数の値が観測されているとき，線形判別関数は式 (2) の形となる．

$$Z = \alpha_1 x_1 + \cdots + \alpha_p x_p \tag{2}$$

ここで，x_i は説明変数，α_i は判別係数と呼ばれる定数である．本論文では，判別値 Z が 0 以上の値をとる場合，fault-prone モジュールであると判定する．

2.3　分　類　木

分類木は，説明変数と目的変数の関係を木構造で表現したモデルである．木の各ノードは 2 つ以上の子ノードを持っており，説明変数の値によっていずれかの子ノードへと分岐する．リーフノードは，いずれかの群に割り当てられている．与えられたケースは，説明変数の値に従ってルートノードから木をたどることで，リーフノードにおいて群の判別が行われる．本論文では，分類木を構築するためのアルゴリズムとして fault-prone モジュール判別で一般的に用いられている CART（Classification And Regression Trees）[7] を用いた．

3. 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．Agrawal らは頻出する組合せ（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，$T_i \in D$ $(1 \leq i \leq n)$，$T_i \subset I$，$I = I_1, \ldots, I_k, \ldots, I_m$（$m$ 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．

各ルールの重要性を表す指標値として次がある．

支持度：支持度は対象データにおける相関ルールの出現頻度であり，$support(X \Rightarrow Y)$ と表記され，$support(X \Rightarrow Y) = s/n$ である．ただし，$s = \left|\{T \in D | X \subset T \cap Y \subset T\}\right|$，$n = \left|\{T \in D\}\right|$．

信頼度：信頼度は前提部 X が満たされたときに同時に結論部 Y も満たされる割合であり，$confidence(X \Rightarrow Y)$ と表記され，$confidence(X \Rightarrow Y) = s/y$ である．ただし，$y = \left|\{T \in D | X \subset T\}\right|$．

リフト値：リフト値は前提部 X によりどのくらい結論部 Y が満たされやすくなっているかを示している．リフト値は 0 より大きい値をとり，リフト値が 1.0 より大きければ大きいほど，前提部 X が含まれることで結論部 Y が満たされやすくなることを表し，逆に，リフト値が 1.0 より小さければ小さいほど，前提部 X が含まれることで結論部 Y が満たされにくくなることを表す．また，1.0 のときは X が Y に寄与しないことを表す．リフト値は $lift(X \Rightarrow Y)$, と表記され，$lift(X \Rightarrow Y) = \frac{confidence(X \Rightarrow Y)}{z/n}$ である．ただし，$z = \left|\{T \in D | Y \subset T\}\right|$．

たとえば，モジュール数 n を 20 とし，X を満たすモジュール数を 10，Y を満たすモジュール数を 8，X と Y を満たすモジュール数を 6 とする．その際，相関ルール $X \Rightarrow Y$ の支持度は 0.3（6/20），信頼度は 0.6（6/10），リフト値は 1.5（$\frac{0.6}{8/20}$）となる．

本論文では，各モジュールのソースコードメトリクスを前提部，fault の有無を結論部として用いる．ただし，ソースコード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．サイクロマティック数の場合，たとえば low $[0, 10)$，medium $[10, 30]$，high $[30, 1/0)$ 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「medium(10 ≤ cyclomatic number < 30) and medium(10 ≤ fan-in < 25) ⇒ fault」のように表される．このルールでは，モジュールのサイクロマティック数が 10 以上で 30 より小さく，かつ，fan-in が 10 以上で 25 より小さいモジュールの fault の有無を予測することができる．

4. 提 案 手 法

4.1　概　　要

本論文は，fault-prone モジュール判別の精度向上を目的として，相関ルール分析とロジスティック回帰分析を組み合わせた fault-prone モジュール判別手法を提案する．提案手法において相関ルール分析とロジスティック回帰分析を組み合わせるうえで，各ルールの重要性を表す指標値（支持度，信頼度，リフト値）を利用する．

本手法では，重要なルール（支持度，信頼度，または，リフト値の大きなルール）が存在する場合はロジスティック回帰分析よりも優先して相関ルール分析によって判別する．存在しない場合は，任意のモジュールを判別できるロジスティック回帰分析によって判別する．

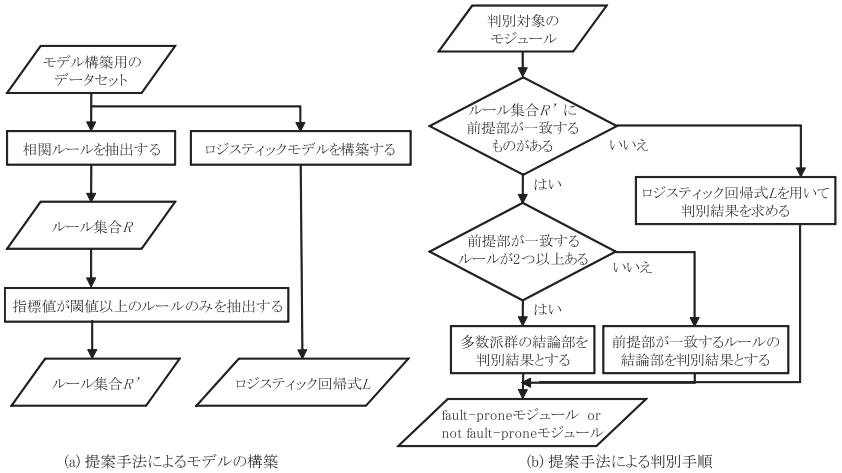

図 1　モデルの構築と判別手順
Fig. 1　Procedure of model building and fault-module prediction.

適用可能な複数のルールが存在する場合には，判別結果の多数決を行う．

ただし，いずれの指標値（支持度，信頼度，リフト値）が本手法に最も適しているのか，さらに，各指標の閾値をどの程度の大きさに設定し，どの程度ルールの絞り込みを行うべきかは明らかではない．上記の点は評価実験（5 章，6 章）により明らかにする．

4.2　モデルの構築と判別の手順

提案手法による，モデルの構築からそのモデルによる判別の流れを**図 1** に示す．まず，図 1 (a) に示すようにフィットデータを用いて，ロジスティック回帰分析によりロジスティック回帰式 L と，相関ルール分析により相関ルールの集合 R とを抽出する．ここで，相関ルール分析をフィットデータに適用するために，あらかじめ量的変数（間隔尺度や順序尺度）を質的変数（順序尺度）に変換し，均等な k 区間に分割する（本論文では，k の値は 5 とした）．たとえば，サイクロマティック数の場合，その最小値が 3 で，最大値が 38 であるとすると，$[3, 10)$，$[10, 17)$，$[17, 24)$，$[24, 31)$，$[31, 38]$ のように，5 つに分割される．

次に，相関ルールの集合 R に対して，指標値があらかじめ与えられた閾値以上のルール（支持度が閾値 $\theta_{support}$ より大きいルール，信頼度が閾値 $\theta_{confidence}$ より大きいルール，もしくは，リフト値が閾値 θ_{lift} より大きいルール）のみを選択する（相関ルール集合 R'）．

そして，図 1 (b) に示すように，判別対象のモジュールのメトリクス値が R' に含まれる相関ルールの前提部に一致するかどうかを判別する．モジュールの判別に相関ルールを用いる場合，前提部が一致するルールの数が 1 つ，もしくは複数個が一致することが考えられる．前提部が一致する相関ルールが 1 つの場合は，その相関ルールの結論部を判別結果とする．複数個ある場合は，多数派群（相関ルール集合 R' に含まれるルールのうち，結論部 Y が多い方の群）の結論部を判別結果とする．R' の中に前提部が一致する相関ルールが存在しない場合は，任意のモジュールを判別可能なロジスティック回帰式 L により判別する．

5.　交差検証法による実験

5.1　概　　要

実験の目的は，各指標（支持度，信頼度，リフト値）の閾値を変化させた場合における，提案手法の判別精度を評価することである．さらに，その判別精度を，3 つの代表的な fault-prone 判別モデル（ロジスティック回帰分析，線形判別分析，分類木）と比較対照することで評価する．本実験では，単一のデータセットをランダムに 2 分割し，一方のデータセット（フィットデータ）を用いて判別モデルを構築し，もう一方のデータセット（テストデータ）を用いてその判別精度を求める（交差検証法）．

支持度，もしくは信頼度を指標値として用いる場合には，各指標の閾値 $\theta_{support}$，もしくは閾値 $\theta_{confidence}$ を 0.1，0.2，0.3,… と 0.1 ずつ変化させた．また，リフト値を指標値として用いる場合には，閾値 θ_{lift} を 1.5 から 2.0，3.0，4.0,… と変化させた．相関ルールの抽出には Morisaki らが開発したプロトタイプツール[12] を用いて，最小支持度は 0.01，相関ルールに含まれる前提部の長さは最大で 4 までとした．

提案手法の比較対象である 3 種類の fault-prone モジュール判別モデル（ロジスティック回帰分析，線形判別分析，分類木）の構築には統計解析用のソフトウェアである R[20] を用いた．

5.2　データセット

実験には，NASA/WVU IV&V facility Metrics Data Program（MDP）[14] が公開しているデータセットのうち，評価実験によく用いられる[9),17)]KC1 のデータセットを用いた．KC1 では，C++で開発されたソフトウェアのモジュールデータが収集されており，ソースコードの規模は 43 k ステップである．KC1 で収集されたデータセットの概要を**表 1** に示す．

実験では，fault の有無を目的変数，21 種類のソースコードメトリクスを説明変数として用いた．ソースコードメトリクスの一覧を**表 2** に示す．

表 1　KC1 で収集されたデータセットの概要
Table 1　Summary of KC1 dataset.

fault あり モジュール数（個）	fault なし モジュール数（個）	fault モジュール 含有率（%）
325	1,782	15.4

表 2　KC1 のソースコードメトリクス
Table 2　Source code metrics of KC1 dataset.

番号	ソースコードメトリクス
m_1	空白の行数
m_2	分岐の数
m_3	コメントを含むコード行数
m_4	コメント行数
m_5	Cyclomatic Complexity
m_6	Design Complexity
m_7	Essential Complexity
m_8	コードの実行数（空行とコメント行以外）
m_9	Halstead 尺度の Content
m_{10}	Halstead 尺度の Difficulty
m_{11}	Halstead 尺度の Effort
m_{12}	Halstead 尺度の Error Estimate
m_{13}	Halstead 尺度の Length
m_{14}	Halstead 尺度の Abstract Level
m_{15}	Halstead 尺度のプログラミング時間
m_{16}	Halstead 尺度の Volume
m_{17}	オペランドの数
m_{18}	オペレータの数
m_{19}	ユニークなオペランドの数
m_{20}	ユニークなオペレータの数
m_{21}	コードの総行数

5.3 実 験 手 順

各指標（支持度，信頼度，リフト値）に対して，以下の手順を行う．

(1)　データセットをランダムに 2 等分し，一方をフィットデータ，もう一方をテストデータとする．

(2)　手順 (1) で作成したデータセットのペアを用いて，提案手法の判別精度を求める．判別の際には，指標値が閾値以上の相関ルールを用いた．

(3)　閾値を変化させて，手順 (2) を繰り返す．

(4)　手順 (1) のデータセットのペアを用いて，比較対象である 3 つの判別モデル（ロジスティック回帰分析，線形判別分析，分類木）の判別精度を求める．

5.4 評 価 指 標

Fault-prone モジュール判別の評価基準として，再現率，適合率，および F1 値[6] を用いた．再現率（Recall）は，fault を含んでいるモジュールのうち，fault を含むモジュールであると正しく判別した割合を表す．また，適合率（Precision）は，fault-prone モジュールと判別したうち，実際に fault を含んでいた割合を表す．

再現率と適合率はトレードオフの関係にあるため，再現率と適合率だけで判別精度を評価することは難しい．そこで，本論文では再現率と適合率の調和平均で与えられる F1 値を評価基準として用いた．F1 値は，式 (3) として定義され，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．

$$F1\text{-}value = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{3}$$

5.5 結　　果

各指標の閾値を変化させた場合の提案手法の判別精度（F1 値）と，比較対象のモデルの判別精度を**図 2** に示す．図 2 の (a)，(b)，(c) に，指標値それぞれの結果を示し，グラフの横軸は各指標の閾値，左側の縦軸は F1 値，右の縦軸は相関ルールが適用できたモジュール（前提部の一致する相関ルールが存在したモジュール）の割合を示す．

図 2 (c) に示すように，提案手法を用いることで，従来手法の中で最も F1 値が大きかった線形判別分析より最大 0.103 向上した．提案手法で用いた指標ごとに以下の傾向が見られた．

- **支持度**　図 2 (a) に示すとおり，閾値 $\theta_{support}$ の変化にかかわらず提案手法の F1 値は，すべての判別手法の中で最も小さい（F1 値はつねに 0）．また，提案手法ではすべてのモジュールが相関ルールによって判別されている，つまり，ロジスティック回帰分析によっていっさい判別されていない．
- **信頼度**　図 2 (b) に示すとおり，閾値 $\theta_{confidence}$ の変化にかかわらず提案手法の F1 値は，支持度を用いた場合と同様，線形判別分析と比較して小さい．閾値 $\theta_{confidence}$ の値が小さい場合，すべてのモジュールが相関ルールによって判別されており，閾値 $\theta_{confidence}$ を 1.0 に設定した場合でも，60%以上のモジュールが相関ルールによって判別されている．
- **リフト値**　図 2 (c) に示すとおり，閾値 θ_{lift} が 3.0 までは，提案手法の F1 値が大きく

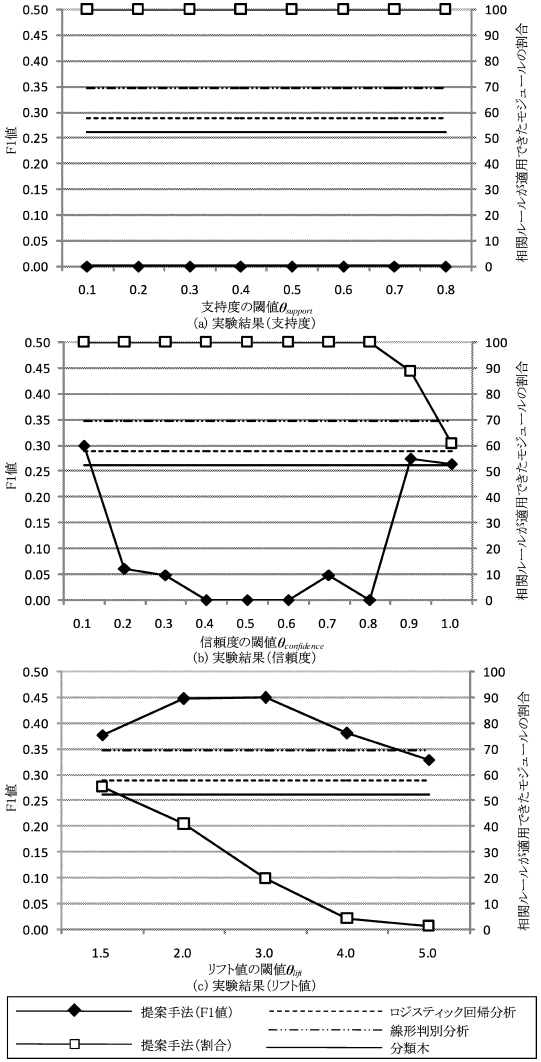

図 2　相関ルール抽出の指標値による判別可能なモジュールの割合と判別精度の変化（KC1）

Fig. 2　Prediction performance of fault-proneness models and percentage of detected modules by rules for different thresholds (KC1).

なり，3.0 からは F1 値が小さくなった．閾値 θ_{lift} の大きさが 1.5 から 4.0 では，提案手法の F1 値は，今回用いた従来手法すべてと比較して大きい．特に，閾値 θ_{lift} が 3.0 のとき（相関ルールによって判別されるモジュールの割合が約 20%のとき），提案手法の F1 値は 0.449，線形判別分析の F1 値は 0.346，ロジスティック回帰分析の F1 値は 0.288，分類木の F1 値は 0.261 であった．一方，閾値 θ_{lift} が 4.0 以上のとき（相関ルールによって判別されるモジュールの割合が 20%より小さい場合），提案手法の性能が低下している．閾値 θ_{lift} が 5.0 のとき（相関ルールによって判別されるモジュールの割合が 5%以下のとき），提案手法の F1 値は線形判別分析と比較して小さい．

結果として，3 つの指標（支持度，信頼度，リフト値）の中で，相関ルール分析とロジスティック回帰分析を組み合わせる際に用いるルール選択には，リフト値が最も適していることが分かった．特に，リフト値 3.0 から 4.0 のとき，提案手法の F1 値は最も大きかった．提案手法の性能向上に寄与していたリフト値 3.0 から 4.0 の相関ルールの一部を**表 3** に示す．表 3 の各行は 1 つの相関ルールを示し，「FP」は結論部が fault-prone モジュールであることを示す．今回の実験では，表 3 に示す相関ルールのように，複雑さ，および，規模を表すメトリクスが大きいという条件が前提部に含まれる場合に，（含まれない場合と比べて高い確率で）fault-prone モジュールであることを示す結果が得られた．また，R_1，R_2，R_3 の前提部に Halstead 尺度の Abstract Level（m_4）が小さい（値が小さいほど，ソースコードが複雑であることを表す）という条件が含まれた．この結果は Halstead 尺度の Abstract Level が fault モジュールの判別に最も寄与するメトリクスであるという報告[15] と一致する．

本実験により得られたロジスティック回帰モデルの概要を**表 4** に示す．ただし，各変数の値域はそれぞれ大きく異なるので，各変数間の値域の差による影響を取り除くため，ここでは各変数を平均 0，分散 1 にして求めた標準化偏回帰係数を示す．有意差（$p < 0.10$）が認められた変数は表 4 のとおりである．

5.6 考　　察

支持度，もしくは信頼度を指標として用いた場合，提案手法の判別精度は従来手法よりも低かった．Song ら[19] は，同時に発生しやすい fault の種類を予測するために，支持度を用いてルールの順位付けを行っているが，我々の提案手法においては判別性能の向上に寄与しなかった．これは，支持度はルールの出現頻度を表す指標であり，前提部が満たされたときに結論部が満たされる確率を表すための指標ではないためであると考えられる．一方，信頼度は，前提部が満たされたときに結論部が満たされる確率を表す指標であるが，fault ありモジュールがフィットデータに含まれる割合が考慮されていないため，信頼度も提案手法の

表 3　KC1 から抽出した相関ルールの一部
Table 3　Example of mined rules in KC1.

番号	前提	結論	支持度	信頼度	リフト値
R_1	$(4.00 \leq m_2) \wedge (m_4 < 0.06) \wedge (17.00 \leq m_5) \wedge (13.00 \leq m_6)$	FP	0.05	0.56	3.47
R_2	$(7.00 \leq m_1) \wedge (m_4 < 0.06) \wedge (17.00 \leq m_5) \wedge (13.00 \leq m_6)$	FP	0.05	0.56	3.47
R_3	$(7.00 \leq m_1) \wedge (4.00 \leq m_2) \wedge (24.00 \leq m_3) \wedge (m_4 < 0.06)$	FP	0.05	0.53	3.26

m_1：分岐の数　m_2：サイクロマティック数　m_3：コードの実行行数（空行とコメント行以外）
m_4：Halstead 尺度の Abstract Level　m_5：ユニークなオペランドの数　m_6：ユニークなオペレータの数

表 4　KC1 のロジスティック回帰モデルの標準化偏回帰係数
Table 4　Standardized partial regression coefficient of logistic model in KC1.

メトリクス	係数	p 値
切片	2.037	0.000
m_3：コメント含むコード行数	0.286	0.057
m_6：Design Complexity	−0.928	0.069
m_{10}：Halstead 尺度の Difficulty	−0.859	0.092

判別性能の向上には寄与しなかったと考えられる．一般的に，モジュールデータは 2 群のケース数に偏りがあることが多い（データセット中の fault ありモジュールと fault なしモジュールの数に大きな差がある）．本データセットの fault ありモジュールの割合も 15.4%であり，2 群のケース数に偏りがあった．そのため，たとえば，“xxx ⇒ fault-prone” というルールにとって信頼度 0.7 は重要であるが，“xxx ⇒ not fault-prone” というルールにとっては重要ではない．その結果，信頼度もまた性能向上に寄与しなかったと考えられる．

一方，リフト値を指標とした場合には，図 2 (c) に示すとおり，提案手法の判別精度は従来手法よりも高かった．また，閾値 θ_{lift} にかかわらず，提案手法の判別精度はロジスティック回帰分析よりも高かった．これは，リフト値によって選択された相関ルールがロジスティック回帰モデルの性能向上に寄与していることを示唆している．ただし，閾値 θ_{lift} を 5.0 に設定した場合のみ，従来手法よりも低くなった．相関ルールによって判別されたモジュールの割合に着目すると，閾値 θ_{lift} を 5.0 に設定した場合，その割合は 1.23%であった．このことから，ほとんどのモジュールが相関ルールによって判別されず，ロジスティック回帰分析によって判別されており，モデルを組み合わせることによる利点が活かされていないことがうかがえる．

表 4 に示すロジスティック回帰モデルの変数と係数からは，fault-prone モジュールに至る要因を知ることが困難であった．その理由として，ソースコードメトリクスは多かれ少なかれ互いに相関があり，完全に独立とはいえないためであると考えており，今後の検討が必要である．

6. 複数バージョン間における実験

6.1 概　　要

複数バージョン間における実験では，fault-prone モジュールを利用する状況により近い条件での実験を行う．そこで，あるバージョンで発見された fault の履歴を基に，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．本論文では Gyimothy ら[5] の方法を用いて Eclipse プロジェクトからモジュールのメトリクス値と fault の履歴を取得した．

本実験において比較する fault-prone モジュール判別手法と評価指標は，5 章と同様である．また，相関ルール分析の条件も 5 章と同様である．ただし，相関ルール分析を行ったところ，最もリフト値の大きい相関ルールであってもその値は 3.0 未満であった．そのため，リフト値を指標として用いる場合，閾値 θ_{lift} の大きさを 1.5，2.0，以降，2.5，2.7，2.9 と変化させて，提案手法の判別精度を求めた．

6.2 データセット

6.2.1 対象プロジェクト

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 1 つである Eclipse の開発中に収集された各メトリクス値を記録したものである．Eclipse のバージョン 3.0（フィットデータ）と 3.1（テストデータ）を実験対象とした．

Eclipse プロジェクトでは，5 章で対象とした KC1 プロジェクトのような整形済みのデータセットが公開されていない．本論文では，以降に示す方法で，各モジュールのメトリクス値と fault の有無を求めた．ここで，モジュールとは，1 つの Java ファイル（拡張子が .java のもの）とする．

表 5　Eclipse のソースコードメトリクス
Table 5　Source code metrics for Eclipse project.

番号	ソースコードメトリクス
m_1	コードの実行数
m_2	メソッドの総行数
m_3	最大ネスト数
m_4	メソッドのパラメータ数
m_5	Cyclomatic Complexity
m_6	フィールド数
m_7	メソッド数
m_8	オーバライドしたメソッド数
m_9	サブクラスの数
m_{10}	静的フィールドの数
m_{11}	静的メソッドの数
m_{12}	SIX（Specialization Index）
m_{13}	DIT（Depth of Inheritance Tree）
m_{14}	LCOM（Lack of Cohesion of Methods）
m_{15}	WMC（Weighted Methods per Class）

表 6　Fault の発見/修正履歴の収集条件
Table 6　Condition for collecting fault reported/fixed archive.

Classification	Eclipse
Product	Platform
Status of faults	Resolved, Verified, Closed
Resolution of faults	Fixed
Severity	Enhancement 以外
Priority	all

6.2.2　メトリクス値の計測

各バージョンのソースコードを取得し，Eclipse Metrics plugin[4] を用いて，各モジュールのメトリクス値を計測した．実験では，fault の有無を目的変数，Eclipse Metrics plugin で収集可能な 14 種類のメトリクスに，コードの実行行数を加えた計 15 種類のソースコードメトリクスを説明変数として用いた（**表 5**）．

6.2.3　fault の発見/修正履歴の収集

メトリクス値を計測した各バージョンのモジュールが fault を含むかどうかを調べるために，Eclipse プロジェクトが運用している障害管理ツール Bugzilla[⋆1]から，**表 6** に示す条件に基づいて，fault の報告と修正の履歴を収集した．

⋆1 https://bugs.eclipse.org/bugs/

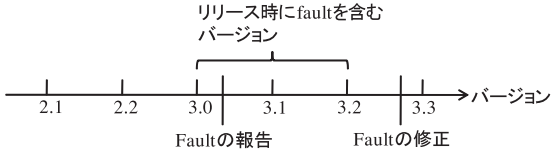

図 3　fault の修正/報告とモジュールが関連する Eclipse のバージョン
Fig. 3　Eclipse version where a bug is associated with a module.

表 7　Eclipse で収集されたデータセットの概要
Table 7　Summary of datasets in Eclipse.

バージョン	fault あり モジュール数（個）	fault なし モジュール数（個）	fault モジュール 含有率（%）
Eclipse 3.0	793	3,659	17.8
Eclipse 3.1	859	4,415	16.3

Fault の報告と修正がどのモジュールに対して行われたものなのかを特定するために，履歴から，モジュール名を取得した．また，どの時期に fault が含まれていたのかを特定するために，fault が報告された日と，fault が修正された日を取得した．

6.2.4　モジュールと fault の有無の関連付け

本論文では，Gyimothy ら[5] が行った方法で，各バージョンの各モジュールに fault が含まれているかどうかを求めた．

まず，どのモジュールに fault が含まれているかを特定した．Bugzilla において，fault の報告と修正の履歴には，fault が修正されたソースコード（修正パッチ）が添付されており，そこにモジュール名が記述されている．そこで，修正パッチに書かれているモジュール名を取得し，どのモジュールに fault が含まれていたのかを特定した．

次に，fault が含まれている時期（バージョン）を特定した．fault が報告された日からその fault が修正されるまでの期間に修正されたモジュールには，fault が含まれているものと見なした．**図 3** に示すように，たとえば，バージョン 3.0 から 3.1 の間に fault が報告され，3.2 から 3.3 の間に修正されたモジュールの場合，バージョン 3.0 から 3.2 の間までの期間，fault を含んでいると見なす．

上記の手順で Eclipse プロジェクトから収集されたデータセットの概要を**表 7** に示す．表 7 に示すとおり，バージョン 3.0 には 4,452 個のモジュールが，バージョン 3.1 には 5,274 個

のモジュールが含まれていた．それぞれのデータセットに fault ありモジュールが含まれていた割合は，17.8%と 16.3%であった．

6.3 実験手順

各指標（支持度，信頼度，リフト値）に対して，以下の手順を行う．

(1) Eclipse のバージョン 3.0 と 3.1 から収集されたデータセットを用いて，提案手法の判別精度を求める．判別の際には，ある指標が閾値以上の相関ルールを用いた．

(2) 閾値を変化させて，手順 (1) を繰り返す．

(3) 手順 (1) と同様のデータセットを用いて，比較対象である 3 つの判別モデル（ロジスティック回帰分析，線形判別分析，分類木）の判別精度を求める．

6.4 結　　果

各指標の閾値を変化させた場合の提案手法の判別精度（F1 値）の変化と，比較対象のモデルの判別精度を**図 4** に示す．図 4 (c) に示すように，提案手法を用いることで，従来手法の中で最も F1 値が大きかった線形判別分析より F1 値が最大 0.163 向上した．提案手法で用いた指標ごとに以下の特徴が見られた．

- **支持度**　図 4 (a) に示すとおり，5 章と同様の結果で，閾値 $\theta_{support}$ の変化にかかわらず提案手法の F1 値は従来手法の中で最も F1 値が大きかった線形判別分析と比較して小さい（F1 値はつねに 0）．また，提案手法では，すべてのモジュールが相関ルールによって判別されている．
- **信頼度**　図 4 (b) に示すとおり，5 章の結果と異なり，閾値 $\theta_{confidence}$ の変化にかかわらず提案手法の F1 値は線形判別分析と比較して小さい（F1 値はつねに 0）．また，提案手法では，すべてのモジュールが相関ルールによって判別されている．
- **リフト値**　図 4 (c) に示すとおり，5 章と同様の結果で，閾値 θ_{lift} の変化にかかわらず提案手法の F1 値は，3 つの従来手法と比較して大きい（閾値 θ_{lift} が 2.5 のとき，提案手法の F1 値は 0.432，従来手法の中で最も精度が良かった線形判別分析の F1 値は 0.269 であった）．

  閾値 θ_{lift} が 2.5 までは，提案手法の F1 値が大きくなり，2.5 からは F1 値が小さくなった．相関ルールによって判別されるモジュールの割合が 20%より小さい場合，提案手法の性能が低下している．

提案手法の性能向上に寄与していたリフト値 2.5 から 2.7 の相関ルールの一部を**表 8** に示す．5 章の実験で得られた結果と同様，複雑さ，および，規模を表すメトリクスが大きい場合に，（そうでない場合と比べて高い確率で）fault-prone モジュールであることを示す結

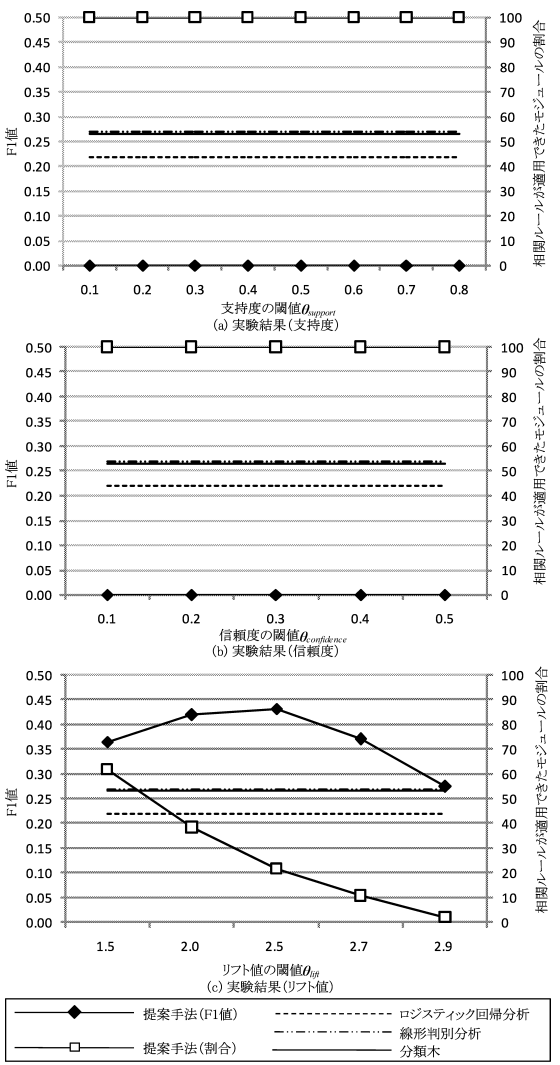

図 4　相関ルール抽出の指標値による判別可能モジュール割合と判別精度の変化（Eclipse）

Fig. 4　Prediction performance of fault-proneness models and percentage of detected modules by rules for different thresholds (Eclipse).

表 8　Eclipse から抽出した相関ルールの一部
Table 8　Example of mined rules in Eclipse.

番号	前提	結論	支持度	信頼度	リフト値
R_1	$(40.00 \leq m_5) \wedge (3.00 \leq m_{13}) \wedge (5.00 \leq m_6) \wedge (1.00 \leq m_8)$	FP	0.01	0.48	2.69
R_2	$(40.00 \leq m_5) \wedge (5.00 \leq m_6) \wedge (2.00 \leq m_{10}) \wedge (181.00 \leq m_1)$	FP	0.03	0.46	2.60
R_3	$(4.00 \leq m_3) \wedge (5.00 \leq m_6) \wedge (2.00 \leq m_{10}) \wedge (181.00 \leq m_1)$	FP	0.02	0.46	2.59

m_1：コードの実行行数　m_3：最大ネスト数　m_5：サイクロマティック数　m_6：フィールド数
m_8：オーバライドしたメソッド数　m_{10}：静的フィールドの数　m_{13}：DIT (Depth of Inheritance Tree)

表 9　Eclipse のロジスティック回帰モデルの標準化偏回帰係数
Table 9　Standardized partial regression coefficient of logistic model in Eclipse.

メトリクス	係数	p 値
切片	1.708	0.000
m_1：コードの実行行数	−0.679	0.064
m_3：最大ネスト数	−0.332	0.000
m_4：メソッドのパラメータ数	0.736	0.000
m_6：フィールド数	0.147	0.016
m_7：メソッド数	−0.679	0.000
m_{10}：静的フィールド数	−0.267	0.019
m_{11}：静的メソッド数	−0.534	0.001
m_{14}：LCOM	−0.223	0.000
m_{15}：WMC	0.699	0.010

果であった．

本実験により得られたロジスティック回帰モデルの概要を**表 9** に示す．5 章と同様，各変数の値域はそれぞれ大きく異なるので，各変数間の値域の差による影響を取り除くため，ここでは各変数を平均 0，分散 1 にして求めた標準化偏回帰係数を示す．有意差（$p < 0.10$）が認められた変数は表 9 のとおりである．表 9 に示すロジスティック回帰モデルの変数と係数からは，5 章と同様，fault-prone モジュールに至る要因を知ることが困難であった．

6.5 考　　察

5 章と同様に，支持度，もしくは信頼度を指標として用いた場合，提案手法の判別精度は従来手法よりも低かった．閾値の大きさにかかわらず，提案手法の F1 値はつねに 0 であった．

一方，リフト値を指標とした場合には，提案手法の判別精度は従来手法よりも高かった．ただし，閾値 θ_{lift} を 2.9 に設定した場合のみ，従来手法とほぼ同じ F1 値の大きさとなった．相関ルールによって判別されたモジュールの割合に着目すると，閾値 θ_{lift} を 2.9 に設定した場合，その割合は 2.03%であった．このことから，ほとんどのモジュールが相関ルールによって判別されず，ロジスティック回帰分析によって判別されており，組合せの恩恵を受けられていないことがうかがえる．現時点では，5 章と本章の結果からは，相関ルールによって判別されるモジュールの割合が 20%程度の高さを保つように閾値 θ_{lift} の大きさを設定することで，組合せの恩恵を受けられると示唆される．また，リフト値 2.0 を閾値としてルールを採用した場合は 5 章，6 章ともに従来手法より F1 値が大きいことから，リフト値 2.0 以上のルールを採用することが fault-prone モジュール判別における 1 つの目安となると考える．

7. 関 連 研 究

ソフトウェアモジュールから計測したソースコードメトリクス（サイクロマティック数，変更行数など）に基づいて，モジュールの fault の有無を推定する fault-prone モジュール判別モデルが多数提案されており[2),3),7),13),16),19)]，ロジスティック回帰分析，線形判別分析，分類木などのモデル化手法が用いられる．たとえば，ロジスティック回帰分析は，与えられた説明変数の値の組に対する，ある現象の発生する条件付き確率（モジュールが fault を含む確率）をロジスティック回帰式としてモデル化する手法である．一般に，fault の混入は確率的に生じるため，確率事象としてとらえることが妥当であるとの考えから，ロジスティック回帰分析は広く用いられてきた[2),3),13)]．

一方で，従来から広く用いられているソースコードメトリクスではなく，ネットワークメトリクス（モジュール間の関係を表すメトリクス）を判別モデルのパラメータとして利用することが Zimmermann らによって提案されている[22)]．重回帰分析，ロジスティック回帰分析を判別モデルとして，Windows Server 2003 を対象とした実験の結果，ネットワークメトリクスをソースコードメトリクスと併用して用いることはソースコードメトリクスを単体で用いるよりも，fault-prone モジュール判別に効果があることを示した．ネットワークメ

トリクスを本論文で提案する手法によって用いることで，さらなる精度向上が期待できる．

また，ソースコードからメトリクスを抽出することなく，ソースコードのテキスト自身を入力として，fault-prone モジュールを判別する手法が Mizuno らによって提案されている[11]．Mizuno らは，メールのスパムフィルタなどに利用されているベイジアンフィルタを用いることで，ソースコードを単なるテキストと見なして fault-prone モジュールの判別を試みている．オープンソースプロジェクトから収集したデータセットを対象とした実験では，fault を含むモジュールの全体のうち 78%のモジュールを検出することができた．本論文で提案する相関ルール分析との組合せを行った場合，ソースコードメトリクスを抽出する必要がないという Mizuno らの手法の利点が失われることにはなるが，判別精度が向上する可能性がある．

ソフトウェア工学以外の分野では，相関ルール分析は，小売販売店での販売履歴を対象としたあわせ買いの傾向の抽出[1]，ウェブサイトのアクセスログを対象とした個々のウェブページのキャッシング，先読みの決定[21]，外膜タンパク質の予測[18] をはじめとして，様々な分野でその効果が認められている．たとえば，販売履歴からのあわせ買いの分析で「商品 A を購入する ∧ 商品 B を購入する ⇒ 商品 C を購入する」というルールが得られた場合，商品 A，B，C を近くに配置し，顧客がそれらの商品の存在を認識しやすくするなどの利用方法が考えられる．

ソフトウェア工学分野における利用例も少ないながらいくつか報告されている．Song ら[19] は，開発中の fault の報告履歴（fault 混入の原因，修正工数など）から相関ルールを抽出し，同時に起こりやすい fault の種類の発見や，fault の修正工数（“1 人時未満”，“1 人時以上，1 人日未満”，“1 人日以上，3 人日未満”，“3 人日以上”）の予測に用いた．浜野ら[23] は，ソフトウェア開発プロジェクトを混乱（開発予算，もしくは開発期間が基準値を超えている状態）に陥らせる要因を明らかにするために，開発プロジェクトのリスク要因に関するメトリクス値を集め，相関ルール分析を行った．Michail[10] は，アプリケーション内でのクラスライブラリの再利用パターンを相関ルール分析を用いて発見し，そのパターンをライブラリ構築時に利用することを試みた．これらの研究は相関ルール分析を単独で利用している一方，我々の研究は fault-prone モジュールの判別精度向上のために相関ルール分析をモデルベースの判別手法と組み合わせている．

8. おわりに

本論文では，fault-prone モジュール判別の精度向上を目的として，相関ルール分析とロジスティック回帰分析を組み合わせた fault-prone モジュール判別手法を提案し，その性能を実験的に評価した．実験により得られた主な結果および知見は以下のとおりである．

- 提案手法を用いることで，3 つの従来手法（ロジスティック回帰分析，線形判別分析，分類木）と比較して F1 値が最大 0.163 向上した．
- 今回用いたデーセットでは，相関ルール分析とロジスティック回帰分析を組み合わせる際に用いる相関ルールの選択には，リフト値が適していることが分かった．また，モデル利用者は，判別されるモジュールの割合が 20%程度の高さを保つように閾値 θ_{lift} の大きさを設定することで，組合せの恩恵を受けられると示唆された．一方，支持度と信頼度は適していないことが分かった．
- リフト値を提案手法に用いた場合には，サイクロマティック数や分岐の数などの複雑さを表すメトリクス値が一定の値を超えると，fault を含む確率が高いことを示す相関ルールが含まれていた．

ただし，これらの結果は 2 つのデータセットから得られたものであり，結果の信頼性を向上させるためには他のデータセットを用いて実験を追加する必要がある．そして，相関ルール分析で正しく判別できなかったモジュールの傾向，ロジスティック回帰分析で正しく判別できなかったモジュールの傾向について分析することも今後の課題の 1 つである．

なお，参考までに線形判別分析や分類木を相関ルールの組合せに用いた場合の結果を次に示す．紙面の都合上，5 章，6 章の実験のうち，本実験で最も F1 値が大きくなったリフト値 $\theta_{lift} = 3.0$，$\theta_{lift} = 2.5$ の場合の結果のみを示す．線形判別分析，分類木それぞれと組み合わせた場合の F1 値は，5 章のデータセットを用いた場合では 0.454，0.441 であり，6 章のデータセットでは 0.433，0.432 であった．いずれの場合においても，相関ルールとの組合せによって精度の向上が得られた．

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．

参考文献

1) Agrawal, R., Imieliński, 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) Basili, V.R., Briand, L.C. and Melo, W.L.: A Validation of Object-Oriented Design Metrics as Quality Indicators, *IEEE Trans. Softw. Eng.*, Vol.22, No.10, pp.751–761

(1996).
3) 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. Softw. Eng.*, Vol.19, No.11, pp.1028–1044 (1993).
4) Frank Sauer: Eclipse Metrics plugin. http://sourceforge.net/projects/metrics
5) Gyimothy, T., Ferenc, R. and Siket, I.: 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).
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) Khoshgoftaar, T.M. and Allen, E.B.: Modeling Software Quality with Classification Trees, *Recent Advances in Reliability and Quality Engineering*, pp.247–270, World Scientific, Singapore (1999).
8) 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* (*ICSE'06*), pp.413–422 (2006).
9) Matsumoto, S., Kamei, Y., Monden, A. and Matsumoto, K.: Comparison of Outlier Detection Methods in Fault-Proneness Models, *Proc. 1st Int'l Symposium on Empirical Software Engineering and Measurement* (*ESEM'07*), pp.461–463 (2007).
10) Michail, A.: Data mining library reuse patterns using generalized association rules, *Proc. 22nd Int'l Conf. on Software Engineering* (*ICSE'00*), pp.167–176 (2000).
11) 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).
12) Morisaki, S., Monden, A., Tamada, H., Matsumura, T. and Matsumoto, K.: Mining Quantitative Rules in a Software Project Data Set, *IPSJ Journal*, Vol.48, No.8, pp.2725–2734 (2007).
13) 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).
14) NASA/WVU IV&V Facility: Metrics Data Program. http://mdp.ivv.nasa.gov/
15) NASA/WVU IV&V Facility: Metrics Data Program—Halstead Metrics. http://mdp.ivv.nasa.gov/halstead_metrics.html#L
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) Seliya, N. and Khoshgoftaar, T.M.: Software quality estimation with limited fault data: A semi-supervised learning perspective, *Software Quality Journal*, Vol.15, No.3, pp.327–344 (2007).
18) She, R., Chen, F., Wang, K., Ester, M., Gardy, J.L. and Brinkman, F.S.L.: Frequent-subsequence-based prediction of outer membrane proteins, *Proc. ACM SIGKDD Int'l Conf. of Knowledge Discovery and Data Mining*, pp.436–445 (2003).
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) The R Foundation: R. http://www.r-project.org/
21) Yang, Q., Zhangand, H.H. and Li, T.: Mining Web Logs for Prediction Models in WWW Caching and Prefetching, *Proc. ACM SIGKDD Int'l Conf. of Knowledge Discovery and Data Mining*, pp.473–478 (2001).
22) Zimmermann, T. and Nagappan, N.: Predicting defects using network analysis on dependency graphs, *Proc. 30th Int'l Conf. on Software Engineering* (*ICSE'08*), pp.531–540 (2008).
23) 浜野康裕，天嵜聡介，水野　修，菊野　亨：相関ルールマイニングによるソフトウェア開発プロジェクト中のリスク要因の分析，コンピュータソフトウェア，Vol.24, No.2, pp.79–87 (2007).
24) 亀井靖高，森崎修司，門田暁人，松本健一：相関ルール分析とロジスティック回帰分析を用いた fault-prone モジュール予測手法の提案，ソフトウェア工学の基礎 XIV，日本ソフトウェア科学会（FOSE2007），pp.125–130 (2007).

(平成 20 年 4 月 14 日受付)
(平成 20 年 9 月 10 日採録)

亀井　靖高（学生会員）

平成 17 年関西大学総合情報学部卒業．平成 19 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．現在，同大学博士後期課程在籍．平成 20 年日本学術振興会特別研究員（DC2）採用，現在に至る．エンピリカルソフトウェア工学，特にソフトウェア信頼性の研究に従事．電子情報通信学会，IEEE 各会員．

森崎　修司（正会員）

平成 13 年奈良先端科学技術大学院大学情報科学研究科博士課程修了．同年株式会社インターネットイニシアティブ入社．オンラインストレージサービスの企画/開発管理，RFID ソフトウェアの国際標準策定活動に従事．平成 17 年奈良先端科学技術大学院大学研究員．平成 19 年同大学情報科学研究科助教．博士（工学）．エンピリカルソフトウェア工学，インターネットを介した知識共有の研究に従事．IEEE 会員．

門田　暁人（正会員）

平成 6 年名古屋大学工学部電気学科卒業．平成 10 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年同大学同研究科助手．平成 16 年同大学助教授．平成 19 年同大学准教授．平成 15〜16 年 Auckland 大学客員研究員．博士（工学）．ソフトウェアメトリクス，ソフトウェアプロテクション，ヒューマンファクタの研究に従事．電子情報通信学会，日本ソフトウェア科学会，教育システム情報学会，IEEE，ACM 各会員．

松本　健一（正会員）

昭和 60 年大阪大学基礎工学部情報工学科卒業．平成元年同大学大学院博士課程中退．同年同大学基礎工学部情報工学科助手．平成 5 年奈良先端科学技術大学院大学助教授．平成 13 年同大学教授．工学博士．エンピリカルソフトウェア工学，特に，プロジェクトデータ収集/利用支援の研究に従事．電子情報通信学会，日本ソフトウェア科学会，ACM 各会員，IEEE Senior Member．