Vol.2011-SE-171 No.20
2011/3/15

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[†1]　門田 暁人[†1]
森崎 修司[†1]　松本 健一[†1]

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線形判別分析やロジスティック回帰分析を始めとする 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，数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．そこで本稿では，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．相関ルール分析で作成されるルールは，バグを含む条件が明確なので，現場に受け入れられやすいという利点がある．ただし，ルールが大量に生成されすぎてしまい，どのルールに着目すればいいのか分からないという点が問題である．そこで，本稿では，相関ルールを絞り込むアルゴリズムを提案し，その効果を実験により評価する．

Reduction of Association Rules for Fault-prone Module Detection

TOMOKI NISHIKAWA,[†1] AKITO MONDEN,[†1]
SHUJI MORISAKI,[†1] and KEN-ICHI MATSUMOTO[†1]

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 intuitive form (cond. ⇒ faulty or cond. ⇒ not faulty). However, although each association rule is enough comprehensive, usually too many (similar) rules are extracted by the association rule mining. This paper proposes a rule reduction method that can eliminate complex (long) rules as much as possible without reducing the prediction performance.

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，テスト工程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 8). 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 2)3)4)5)6)7) ．代表的なモデルとして，ロジスティック回帰分析，線形判別分析，サポートベクターマシン，ランダムフォレストなどが用いられている 6)7)8)9)10)11) ．

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．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を用いた fault-prone モジュール判別 12)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．

ただし，相関ルール分析では，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，ルールを絞り込む方法がよく用いられているが，この方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）ルールのみを残す方法も考えられるが，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．

本稿では，判別精度をできるだけ落とさずに，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．提案方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．

以降，2 章で相関ルール分析の概要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．

[†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

Vol.2011-SE-171 No.20
2011/3/15

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．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，与えら得えられた頻度 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．

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，$\mathrm{support}(X \Rightarrow Y)$ と表記され，$\mathrm{support}(X \Rightarrow Y) = s/n$ である．ただし，$s = |\{T \in D \mid X \subset T \cap Y \subset T\}|, n = |\{T \in D\}|$．
- **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，$\mathit{confidence}(X \Rightarrow Y)$ と表記され，$\mathit{confidence}(X \Rightarrow Y) = s/y$ である．ただし，$y = |\{T \in D \mid X \subset T\}|$．
- **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 のときのときは　X が Y に寄与しないことを表す．リフト値は $\mathit{lift}(X \Rightarrow Y)$，と表記され，$\mathit{lift}(X \Rightarrow Y) = \dfrac{\mathit{confidence}(X \Rightarrow Y)}{z/n}$ である．ただし，$z = |\{T \in D \mid Y \subset T\}|$．

本稿では，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更する. 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．

2.2 相関ルール分析の問題点

相関ルール分析の問題は，条件部の一部のみが異なっていたり，条件部で一方のルールがもう一方のルールを包含しているなど，似たルールが大量に生成されてしまう点である．そのため，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．例えば，A∧B∧C∧D⇒X と A∧B∧C∧E⇒X のように，条件部の一部が異なっていたり，A∧B∧C∧D⇒X と A∧B∧C⇒X のように条件部で一方のルールがもう一方のルールを包含している場合などである．また，A∧B∧C∧D∧E∧F⇒X のように条件部が複雑である（長い）場合，やはり開発者による解釈が難しくなり問題である．

従来，支持度，信頼度，リフト値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. 例えば, 信頼度の高いルールのみを残す方法を図 1 に示す．しかし，この従来のルールの絞り込み手法にも問題点があり，ルールを減らすことはできるが，類似するルールが削減されるわけではないという点である．例えば，①A∧B⇒X(信頼度 82%)，②A∧B∧C⇒X(信頼度 83%)，③A∧B∧D∧F∧G⇒X(信頼度 82%)，④F∧G⇒X(信頼度 50%)という 4 個のルールがあったとき，従来手法では，閾値の信頼度は 80%などに設定され，④のルールが削除される．しかし，似たようなルールである①と②の両方が残ってしまい，長いルール③も残ってしまう．そのため，開発者やテスト実施者のルール理解度を下げてしまう結果となる．

この問題点を解決するために，本稿の次章では，判別精度をできるだけ落とさずに，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する．

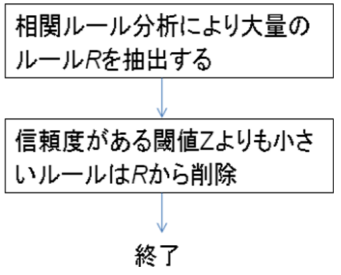

図 1　従来のルールを絞り込むアルゴリズム

Vol.2011-SE-171 No.20
2011/3/15

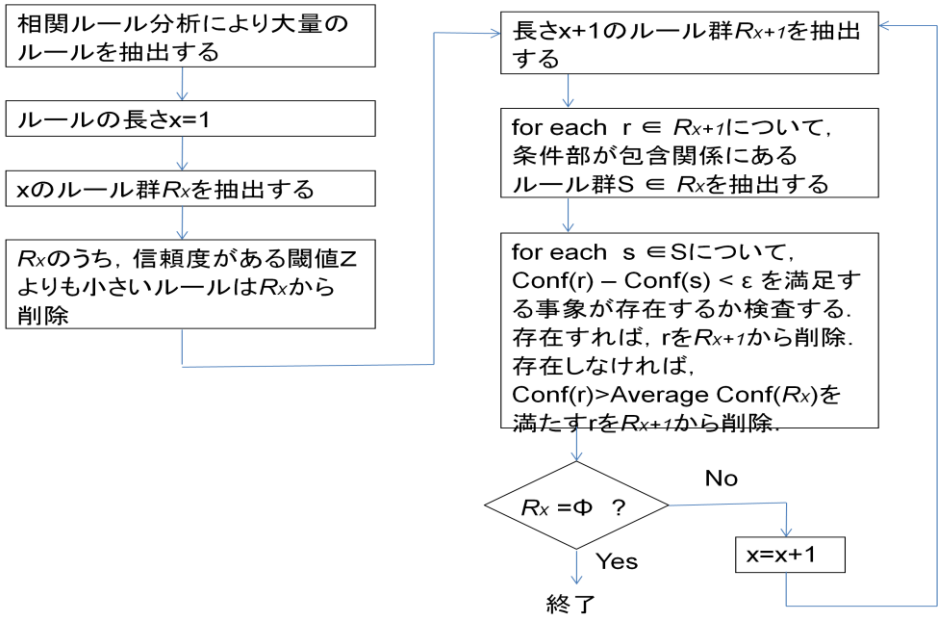

図 2　提案アルゴリズム

3. 提案手法

提案手法では，（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する．例えば，(ルール a) A∧B⇒X，信頼度 82%と（ルール b）A∧B∧C⇒X，信頼度 83%が存在する場合，ルール b は条件∧C 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 b 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．また，仮に，ルール b の信頼度が仮に 95%であった場合，条件∧C によりルール b は長くなっているが，ルール a と比べて信頼度が 13%も高く，判別精度向上のためには残しておくことが望ましいため，削除対象としない．

提案アルゴリズムを図 2 に示す．このアルゴリズムでは，一方が他方のルールの条件を包含する場合，信頼度の差がある閾値より小さい場合に，削除対象とする．図中，Conf(r)はルール r の信頼度を表す．

4. 評価実験

4.1 実験の概要

実験では，従来から用いられているルールを絞り込む手法（以下，従来手法）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．

4.2 実験に用いたデータセット

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 1.0 と 2.0 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．

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．

4.3 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．

手順 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．最大結合数は 4，最小信頼度は 0.01 に設定しておき，足切りしたうえで，ルールを作成する．このとき，バグ有りルールのみを残す．

手順 2. 従来手法の適用 手順 1.で抽出したルールを従来の最小信頼度を閾値とした絞り込み手法を適用し，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．

手順 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に分け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．

手順 4. 閾値の変化 それぞれの閾値を変化させて，手順 2.，手順 3.を繰り返す．

4.4 評価尺度

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．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 3 で示す記号を用いて式(1)で定義される．

$$\text{再現率} = \frac{n_{22}}{n_{21} + n_{22}} \tag{1}$$

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 3 で示す記号を用いて式(2)で定義される．

$$適合率 = \frac{n_{22}}{n_{12} + n_{22}} \quad (2)$$

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，式(3)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．

$$F = \frac{2 \times 再現率 \times 適合率}{再現率 + 適合率} \quad (3)$$

表 1　実験に用いたデータセットの概要

バージョン	fault あり モジュール（個）	fault なし モジュール（個）	fault モジュール 含有率（%）
Mylyn 1.0	425	598	41.5
Mylyn 2.0	663	599	52.5

表 2　ルールの条件部となる実験に用いたデータセットのメトリクス

メトリクス名	メトリクスの種類	内容
TLOC	Code Merics	ソースコード行数
FOUT_total	Code Merics	ファンアウトの総数
MLOC_total	Code Merics	メソッド行の総数
NBD_max	Code Merics	ネストの最大の深さ
PAR_total	Code Merics	パラメータの総数
VG_total	Code Merics	サイクロマティック複雑度の総数
NOF_total	Code Merics	フィールドの総数
NOM_total	Code Merics	メソッドの総数
NSF_total	Code Merics	静的フィールドの総数
NSM_total	Code Merics	静的メソッドの総数
ACD	Code Merics	匿名のタイプ宣言の数
NOI	Code Merics	インターフェースの数
NOT	Code Merics	クラスの数
TPC	Process Metrics	事前変更の総数
BFC	Process Metrics	事前のバグ修正回数
PRE	Process Metrics	リリース前のバグ数

表 3　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}

5. 結果と考察

それぞれの評価指標(F1 値，再現率，適合率)における従来手法と提案手法の結果を図 3 に示す．左から，F1 値，再現率，適合率のグラフで，それぞれの横軸はルールの長さの総数を表している．また，それぞれの評価指標における「全ルールの長さの総数の 0 近傍の拡大図」も図 4 に示す．配置の仕方や横軸は図 3 と同様である．

F1 値に関しては，従来手法よりも提案手法の方が，ルールの削減率における精度の減退率が低い結果が得られた．具体的には，従来手法でルールの長さの総数が 198 のとき F1 値が 0.267 程度であるのに対し，提案手法でルールの長さの総数が 204 のとき F1 値が 0.650 程度と，従来手法と比べて提案手法が大幅に良い結果が見られた．

再現率に関しても，提案手法の方が，ルールの削減率における精度の減退率が低い結果が得られた．具体的には，従来手法でルールの長さの総数が 198 のとき再現率が 0.160 程度であるのに対し，提案手法でルールの長さの総数が 204 のとき再現率が 0.555 程度と，従来手法と比べて提案手法が大幅に良い結果が見られた．

適合率に関しては，F1 値や再現率の結果と比べると，精度の良い結果ではないが，従来手法とさほど遜色はない結果が見られた．具体的には，従来手法でルールの長さの総数が 700 のとき適合率が 0.763 程度であるのに対し，提案手法でルールの長さの総数が 648 のとき適合率が 0.774 程度と，従来手法と比べて遜色はない結果が見られた．

これらの事から，提案手法は，複雑なルールを効率的に除去できるだけでなく，ルールの数という点でも効率的に除去でき，最も必要なルールのみに絞り込むことができる．

Vol.2011-SE-171 No.20
2011/3/15

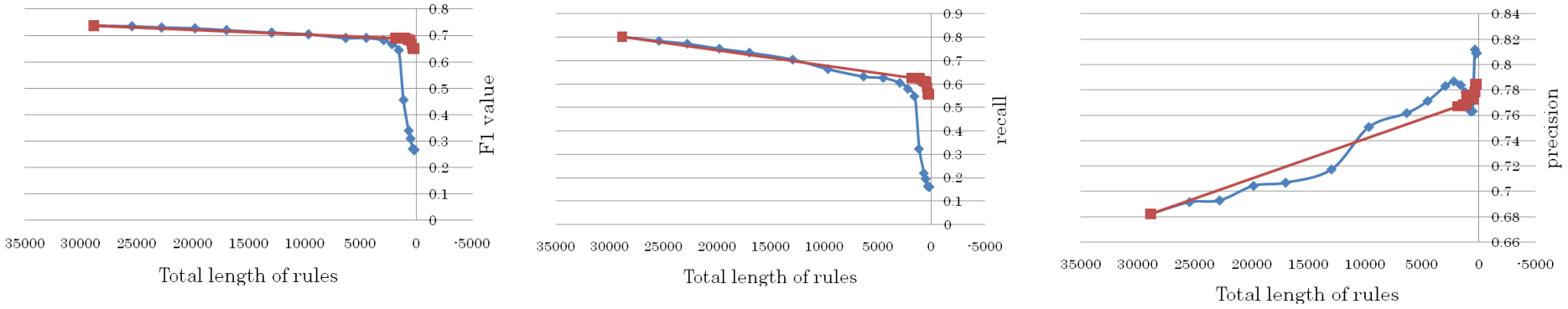

図 3　F1 値，再現率，適合率

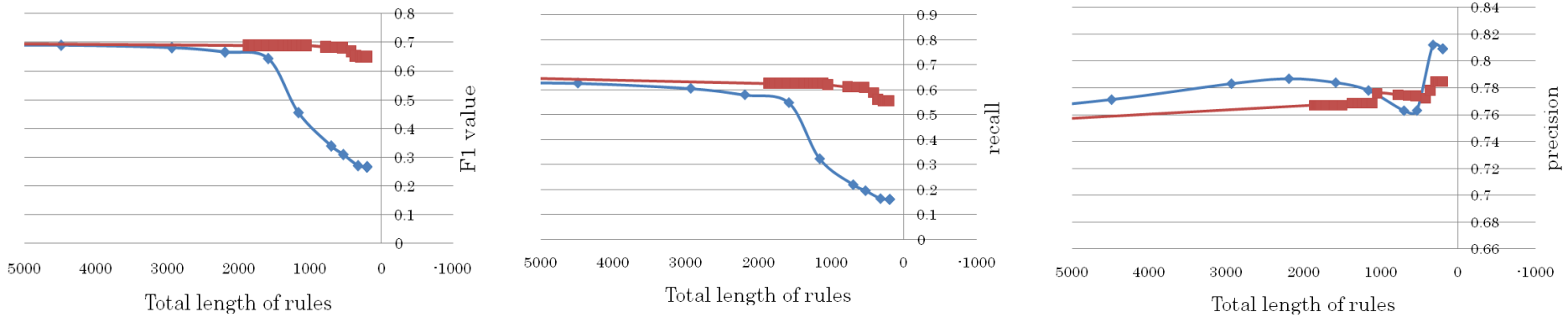

図 4　拡大図

6. おわりに

本稿では，従来の手法よりも効率的にルールを絞り込む手法を提案した．Mylynを用いて，提案手法を実験的に評価した．実験から得られた知見は以下のとおりである．

- F1 値は，従来手法に比べると，提案手法の方が，ルールの長さの総数が少なくなるにつれて，つまり，ルールが分かりやすくなるにつれて，精度の減退率が低い結果が得られた．
- 再現率に関しても，従来手法に比べると，提案手法の方が，精度の減退率が低い結果が得られた．
- 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．

今後の課題として，より多くのデータセットで評価を行うことが挙げられる．

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，本研究の一部は，文部科学省科学研究費補助金（基盤研究(C)：課題番号 22500028）に基づいて行われた．

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) Basili, V.R., Briand, L.C. and Melo, W.L.: A Validation of Object-Oriented Design Metrics as Quality Indicators, *IEEE Trans. Software Engineering*, Vol.22, No.10, pp.751–761 (1996).
3) Gray, A. R. and MacDonell, S. G.: Software Metrics Data Analysis—Exploring the Relative Performance of Some Commonly Used Modeling Techniques, *Empirical Software Engineering*,Vol.4, No.4, pp.297–316 (1999).
4) Gyimothy, T., Ferenc, R. and Siket, I.: Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction, *IEEE Trans. Software Engineering*, Vol.31, No.10, pp.897–910 (2005).
5) 亀井靖高，枩本真佑，柿元 健，門田暁人，松本健一: Fault-proneモジュール判別におけるサンプリング法適用の効果，情報処理学会論文誌, Vol.48, No.8, pp.2651-2662 (2007).
6) 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).
7) 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.
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) 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).
10) Nagappan, N., Ball, T., and Zeller, A.: Mining metrics to predict component failures, *Proc. Int'l Conf. on Software Engineering* (*ICSE'06*), pp.452-461 (2006).
11) Pighin, M. and Zamolo, R.: A Predictive Metric Based on Discriminant Statistical Analysis, *Proc. 19th Int'l Conf. on Software Engineering* (*ICSE'97*), pp. 262- 270 (1997).
12) 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).