トピックを識別したバグ予測モデルの構築

畑 秀明[1,a)]　松本 健一[1,b)]

概要：バグ予測は，テストやレビューの工数を効率よく集中させることができると期待されている．しかし，実際のソフトウェア開発におけるバグ予測適用の難しさの一つとして，予測結果のあいまいさが指摘されている．「バグを含む可能性が高い」という予測結果は，どのような観点のテストを用意すべきかが不明であり，バグ発見までには手間がかかる．この問題に対して，全てのバグを予測するのではなく，脆弱性や影響が大きいと考えられるバグに限定した予測モデルが提案されている．こうした詳細なバグ予測は，バグを細分化することにより予測対象バグの出現する割合が小さくなるため，予測精度が悪くなるという問題がある．具体的には，特に Precision を高くすることが難しくなる．本稿では，構造方程式モデリングで変数間の関係を検討し，ベイジアンネットワークで予測モデルを構築することで，(1)Precision の高い予測結果を得ることができ，また，(2) 高い予測精度が出ないときも問題点を議論できるようになった．

キーワード：バグ予測，バグトピック予測，マルチラベル予測，バグトピック

Building Multi-Label Bug Prediction Models for Bug Topics

Hideaki Hata[1,a)]　Ken-ichi Matsumoto[1,b)]

Abstract: Bug prediction has been considered beneficial since developers and managers can concentrate their quality assurance efforts on bug-prone entities. However, such prediction of future bugs do not offer the details, and such vagueness is one of the major causes of limited adoption of bug prediction in practice. With the limited resource of costs and times, investigating all bug-prone entities is not practical. Taking such practical issue into account, some specific bug prediction models have been proposed. The models focuses on software vulnerabilities, breakage bugs, and bugs in rarely changed entities. The problems of such detailed bug prediction studies exist in the accuracy of prediction results. Since the ratios of targeted bugs are small, obtaining high accuracies become difficult. Achieving high precision is especially difficult. We built bug topic prediction models with Bayesian networks based on the structures obtained from structural equation analysis (SEM). The results show that (i) we can achieve high precision in some cases, and (ii) when we cannot achieve good prediction accuracy, we can discuss the problems.

Keywords: Bug prediction, Bug topic prediction, Multi-label prediction, Bug topics

1. はじめに

バグの含まれる可能性が高いソースコードモジュールを予測する，バグ予測（フォールトプローンモジュール予測とも呼ばれる）は，テストやレビューの工数を効率よく集中させることができると期待され，多くの研究が行われている [1]．バグの有無や密度などを予測するモデルを構築するため，複雑度メトリクス [2,3] やソースコードの変更量 [4]，変更回数や過去のバグ数 [5]，変更の複雑度 [6]，開発者に関するメトリクス [7–9] が計測されてきた．

多くの研究が行われているバグ予測であるが，実際のソフトウェア開発での適用は未だ多くない．国際会議 ESEC/FSE2011 の参加者に対して行われたサーベイでは，適用への障害として，ツールがないこと，予測精度が十分でないこと，モデルの解釈が難しいこと，テストやレビュー

[1] 奈良先端科学技術大学院大学
NAIST, Ikoma, Nara 630–0192, Japan
[a)] hata@is.naist.jp
[b)] matumoto@is.naist.jp

といった活動が適切にモデル化されていないことが挙げられた[*1]．Shihab らは，バグ有りと予測されるコードが多すぎることが問題だと指摘している [10].

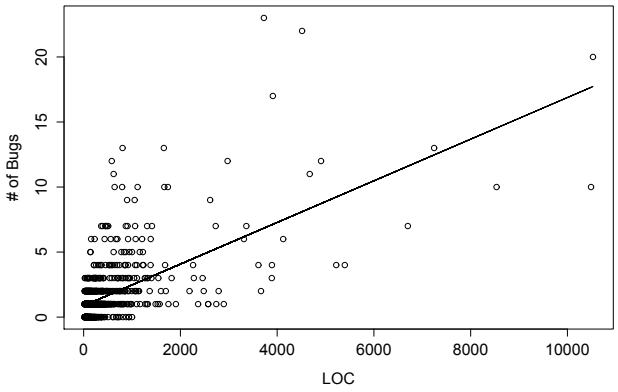
全てのバグを一様に扱うバグ予測は，どんなバグがあるか，また，どう対処すべきかが不明であり，限られた期間に全てのリストに対応することは現実的でないことがある．そこで，特定のバグのみを対象とする予測モデルが提案されている．Shin ら [11] や Zimmerman ら [12] は，セキュリティに関するバグ（脆弱性）に特化した予測モデルを構築している．Shihab らは，影響が大きいと考えられるバグとして，既に存在している機能を破壊するバグ（brekage defects）とあまり変更されていないソースコードから発見されるバグ（surprise defects）を対象とした予測モデルを構築している [10].

このような特定のバグを対象としたバグ予測は，レビューやテスト観点を絞り込みやすいため，実際の開発にも有用と考えられる．しかし，こうした詳細なバグ予測は，バグを細分化することにより予測対象バグの出現する割合が小さくなるため，予測精度が悪くなるという問題がある [10, 13].

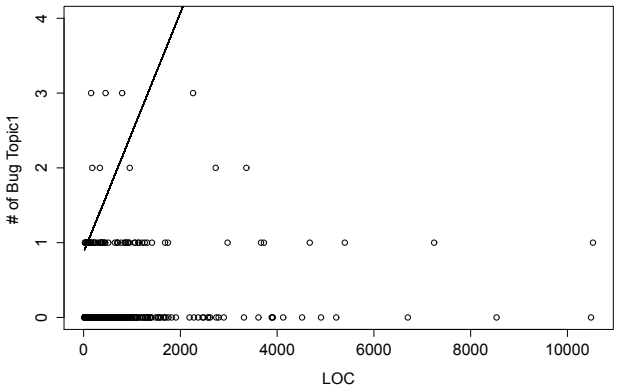
従来のバグ予測では，線形回帰モデルのように均一的な分散や正規誤差を仮定した近似モデルを構築することが多い．しかし，分類されたそれぞれのバグをモデル化する場合にはこれまでの近似が不適切となることが多いことを確認した．そこで，本稿ではポアソン誤差を仮定したモデルを構築する．バグの分類は，大量の文書から文書中のトピックを自動的に推定するトピックモデリング技術を，バグレポートに適用して行う [14–16] ．バグレポートから推定したトピックを**バグトピック**と呼ぶ．モデル構築に際して，計測したメトリクスと，応答変数となるバグトピックとの関係の検討には構造方程式モデリングを用いた．オープンソースソフトウェアへの適用実験から，(1) 高い予測結果を得ることができることを確認し，また，(2) 高い予測精度が出ないときも問題点を議論できるようになった．

2. バグモデルと誤差分散

バグを分類してモデルを構造することの影響を明らかにするために，**図 1** に (a) バグ数と (b) トピック 1 のバグ数（分類したバグの例．トピックについては後述する．）の散布図を示す．簡単のためにコードサイズ（LOC）とバグ数の関係のみを示している．それぞれの図に，線形回帰で得られたモデルを描画した．図 1(a) の直線は，従来分析されてきた，分類なしのバグ数のモデルであり，問題はないように見える．一方，図 1(b) の直線は，分類された特定のバグ数をモデル化したものである．分布に対して適切なモデルとは思われない．



(a) LOC とバグ数（分類なし）の散布図



(b) LOC とトピック 1 の（分類された）バグ数の散布図

図 1 バグ数とトピック 1 のバグ数の散布図
Fig. 1 Plots of the number of bugs/bug topic1 and LOC.

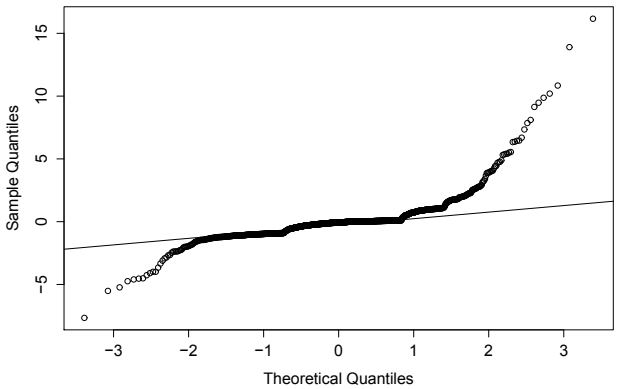


図 2 バグモデル（図 1(b)）の正規 Q-Q プロット
Fig. 2 Normal Q-Q plot of bug model (Fig. 1(b)).

線形回帰モデルでは，誤差の正規分布を仮定する．**図 2** は，図 1(b) の線形回帰の残差の正規 Q-Q プロットである．残差が正規分布に従うときは，その点列は直線上に現れるはずである．点は直線から外れており，残差の正規性につ

[*1] A survey of bug prediction models group in ESEC/FSE2011 PhD Working Groups. http://pwg.sed.hu/sites/pwg.sed.hu/files/PWG1.pdf

表 1 対象プロジェクトの概要
Table 1 Statistics of the Subject Projects.

バージョン	日時	ファイル数	総行数（KLOC）
JDT 3.4	2008-06-13	1,426	465
JDT 3.5	2009-05-26	1,437	477
JDT 3.6	2010-06-03	1,429	482
PDE 3.4	2008-06-03	2,077	323
PDE 3.5	2009-06-02	2,222	363
PDE 3.6	2010-06-01	2,331	389

いて満たされているとはいえない．

バグ数は非負の整数データであるため，ポアソン誤差を仮定した回帰モデルの方がより好ましいと考えられる．ポアソン分布は，平均 λ が大きい場合に正規分布に近似できることが知られている．バグを識別しないモデルの場合，平均バグ数が大きく，誤差を正規分布と仮定したモデルで予測が可能であったと思われる．一方，バグを分類して扱う場合，それぞれの平均バグ数が小さくなる．その結果，バグを識別したモデルでは正規分布を仮定したモデルでは十分な予測が出来ないと考えられる．そこで本稿では，これまで検討されていない，ポアソン誤差を仮定した予測モデルの構築を行う．

3. バグトピック

3.1 トピックモデリング

トピックモデリングは，文書から自動的にトピックを推定する機械学習技術である．トピックとは文書中で何度も共起する単語の集合である．トピックモデリングは自然言語処理の分野で，検索やクラスタリングなど，構造もラベルも無い文書を自動で構造化するために開発された．トピックは，あらかじめ設定したトピック数に合わせて推定，分類される．ソフトウェア開発においても，ソースコードのコメントやバグレポート，メールなど，構造化されていない文書が多数あるため，トピックモデリングのいくつもの適用報告がある [17]．

3.2 対象プロジェクト

本稿で対象とするオープンソースソフトウェアのプロジェクトを示す．Eclipse から，Eclipse JDT Core[*2] と Eclipse PDE UI[*3] の 2 つのプロジェクトを選んだ．これらのプロジェクトは，活発に開発されており，関連するバグレポートと版管理リポジトリが公開されている．以降，それぞれ **JDT**，**PDE** と表記する．

予測モデルの学習データとテストデータとして，それぞれのプロジェクトから 3 つのスナップショットを用いる．いずれのプロジェクトも，バージョン 3.4，3.5，3.6 のスナップショットを選択した．**表 1** に，それぞれのスナップショットの概要となるデータを示す．各バージョン間はおよそ 1 年の開発期間がある．プロジェクト JDT は，ファイル数が 1,400 程度で総行数（LOC）47 万程度，プロジェクト PDE は，ファイル数 2,000 以上で総行数が 37 万程度である．

表 2 Eclipse のバグレポートから取得した 10 のバグトピックと頻出する単語上位 5 つ
Table 2 Top 5 Words of 10 Bug Topics in Eclipse Bug Reports.

Topic	Top words
1	npe, view, api, marker, menu
2	prefer, assist, content, except, help
3	compil, fix, quick, warn, target
4	dialog, select, editor, properti, build
5	resourc, type, caus, histori, link
6	set, page, import, search, prefer
7	javadoc, chang, name, editor, close
8	plugin, test, error, junit, version
9	project, view, method, progress, miss
10	doesnt, type, remov, build, npe

3.3 バグレポートへの適用

本稿では，自動的にバグレポートを分類するために，バグレポートにトピックモデリングを適用し，バグレポートのトピック（バグトピック）を得る．バグレポートにトピックモデリングを適用する前に，ノイズを減らすための一般的な前処理を行う．まず，バグレポートに出現する単語のうち，英語のストップワード（“a”，“the”，“it” など）を除去する．次に，単語の表記ゆれを無くすために語幹に統一する（例えば，“compiling” は “compil” となる）．前処理は，R の *tm* パッケージを用いた．トピックモデリングには，R の *topicmodels* パッケージを用いた．このパッケージではトピックモデリングの技術として，広く用いられている LDA（Latent Dirichlet Allocation）の他に，LDA に相関関係を許可するよう拡張した CTM（Correlated Topic Model）も含まれている．バグレポート中のトピックにも相関関係を認める方が自然であるので，CTM を用いた．

Eclipse のバグレポートは，バグ管理リポジトリの Bugzilla[*4] から収集する．プロジェクト JDT と PDE は，Eclipse 全体のうちサブプロジェクト Eclipse に含まれる．本稿では，このサブプロジェクト Eclipse に関連した，2012 年 12 月 31 日までの全バグレポート 69,134 件を収集した．このバグレポートに対して上述の処理を行う．無関係なサブプロジェクトを除去した，7 万件ほどのバグレポートは，バグトピック推定に十分だと考えている．本稿では，バグトピック数を 10 と設定した．これは，解釈可能で以降の分析にちょうどよいトピック数だと思われる．**表 2** に，得られた 10 のバグトピックにおける頻出する単語の上位 5

[*2] http://www.eclipse.org/jdt/core/
[*3] http://www.eclipse.org/pde/pde-ui/
[*4] https://bugs.eclipse.org/bugs/

つを示す．表中の nep は NullPointerException の略である．例えば，トピック 1 は view や menu の単語を含むので GUI に関連したバグトピック，トピック 8 は test や jnuit の単語を含むのでテストに関連したバグトピックと考えられる．

3.4 ソースコード履歴へのリンク付け

得られたトピックから，各バグレポートにどのトピックが含まれているかが決定できる．本稿では 1 つのバグレポートに 1 つのバグトピックを与える．ソースファイルが過去にどのトピックのバグを修正されたかを得るため，バグレポートとソースファイルを結びつける．このバグレポートとソースファイル間のリンク付けは，通常のバグ予測研究で行われているようにバグ ID を用いる．すなわち，ソースファイルの編集履歴においてコミットメッセージにバグ ID が含まれていた場合に，そのバグ ID を持つバグレポートとリンク付けを行う．

4. メトリクス

予測モデル構築のために以下のメトリクスを計測する．これらは多くのバグ予測研究で有用性が報告されている．

PTi: Previous bug Topic i．以前に修正されたバグトピック i のバグ発生回数．1 つのバグ ID の修正を 1 回と数える．すなわち，同じバグ ID の修正が対象のソースコードで何度も行われた場合も 1 回としている．また，1 つのコミットで複数のバグが修正された場合（1 つのコミットメッセージに複数のバグ ID が含まれている場合），それぞれカウントする．以前にバグがあったソースコードには，同様のバグが再発しやすいのではないかと考えられる．これまでのシングルラベルのバグ予測においては，以前に修正されたバグ数は効果的なメトリクスと考えられている [5]．

LOC: 対象コードの行数（Lines of Code）．サイズが大きいほどリスクが高いと考えられる．多くの複雑度メトリクスが LOC との相関が高いと報告されている [5]．

AddDel: 最初の版と計測対象の版を比較した場合の追加行数と削除行数の和．大きく変更されたソースコードはリスクが高いと考えられる [4]．

AGE: ソースコードの存在日時．新規に作成されたソースコードはバグを含む可能性が高いと考えられる [18]．

HCM: 変更プロセスの複雑度（History Complexity Metic，HCM^{3s} [6]）．シングルラベルのバグ予測でよいメトリクスだと知られている．

本稿はファイルレベルの予測モデルを構築する．上記のメトリクスもファイルレベルで計測する．予測対象となるのは次のメトリクスである．

ATi After bug Topic i．将来発生するバグトピック i のバグ発生回数．PTi と同様に，1 つのバグ ID の発生を 1 回と数える．

5. 予測モデル構築

2 節で議論したように，バグを分類してモデルを構築する場合，誤差を正規分布に近似することは適切でない．そこで，トピックを識別したバグ予測モデルの構築には，誤差をポアソン分布で扱う統計モデリングを適用する [19]．

トピックを識別したバグの予測をする場合，それぞれのバグトピックを応答変数とするモデルを構築する必要がある．すなわち，従来のバグ予測が 1 つの応答変数であるのに対して，バグトピック予測では複数の応答変数を持つこととなる．計測メトリクスと応答変数の多対多の関係を検討することは，手間がかかり，また，パラメータ推定も時間がかかる．そこで，潜在変数（因子）を導入してモデル化する．これによって，類似した傾向を示すメトリクスをまとめ，効率よく因果関係を分析できる．

5.1 Step 1: 因子分析

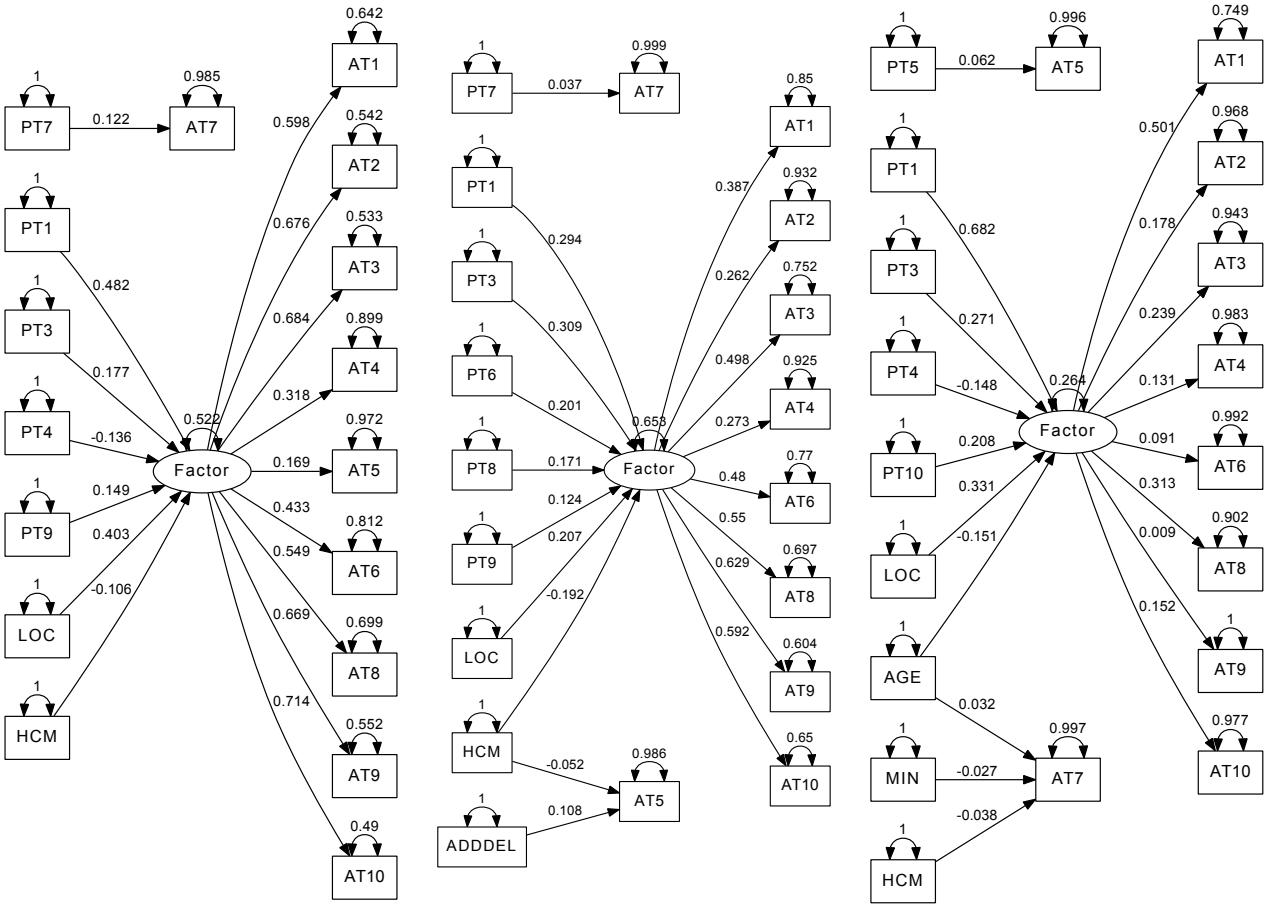
最初に，R の *factonal* パッケージを用いて，予測対象のメトリクス ATi に対して因子分析を行う．因子分析は，観測された変数に隠された因子があると仮定し，その因子の影響を相関で判別する多変量解析の手法である．因子数を事前に与える必要がある．ここで得られた因子を次のステップで用いる．

5.2 Step 2: 構造方程式モデリング

構造方程式モデリング（Structual Equation Modeling，SEM）は，構成概念や観測変数の性質を調べるために，集めた観測変数を同時に分析する統計的手法である．SEM は，因子分析，分散分析，パス分析などを含むといわれている [20]．複雑な因果関係や潜在構造を単一のモデルで表現できる分析手法である．

予測対象メトリクス ATi を，潜在変数を介して，収集したメトリクスから説明するモデルを構造方程式モデリングによって構築する．構造方程式モデリングでは，分析者が仮定した観測変数と潜在変数の因果関係を分析する [20]．そこで，前述の因子分析で得られた因子を導入する．

分析には，R の *sem* パッケージを用いた．構造方程式モデリングは，変数間の関係を表すモデルの記述，パラメータの推定，モデルの改善の手順を繰り返して最終的なモデルを得る．モデルは，観測変数や潜在変数から影響があると思われる独立変数にパスを与えて記述する．最初は，計測メトリクスから潜在変数，また潜在変数から応答変数への全てのパスを与える．パラメータ推定は自動的に行われる．記述が不適切な場合は，エラーが出るので記述を改める．パラメータ推定が完了すると，**図 3** のようなモデルが得られる．2 つの変数間の数値は回帰係数であり，1 つの変数上にある数値は誤差の分散である．誤差の分散が 0 に



(a) PDE 3.4 (b) PDE 3.5 (c) PDE 3.6

図 3 構造方程式モデリングのパス図
Fig. 3 Path diagrams from SEM.

近いほど，変数が他の観測変数や潜在変数で説明できているということなので好ましい．回帰係数の値が小さい（本稿では 0.1 に満たないものとした）パスを除去して，モデルを改善する．sem パッケージはモデルの適合度を示す指標を算出するが，あてはまりのよいモデルを構築することが目的ではないので参考にはしない．余分なパスやメトリクスを除去し，シンプルなモデルが得られたら終了する．

図 3 は，プロジェクト PDE の 3 つのスナップショットで得られたメトリクスを用いて，構築したモデルである．独立に構築したモデルであるが，似た構造をしていることが分かる．例えば，メトリクス AT5 や AT7 は，他のバグトピックのメトリクスとは独立している．他のバグトピックメトリクスは 1 つの潜在変数で説明され，その潜在変数は過去のバグトピックメトリクス PT1 の影響が特に大きい．プロジェクト JDT も，同様に似た構造が得られた．このような因果関係や潜在構造の類似性から，過去のデータから学習したモデルで将来を予測できそうだと期待できる．

5.3 Step 3: 階層ベイズモデル

構造方程式モデリングは，メトリクスと潜在変数間の関係を分析する強力な手法であるが，予測には適さない．この構造方程式モデリングの限界には，ベイジアンネットワークによる補完が有用だといわれている [21,22]．ベイジアンネットワークは，同じ構造を新しいデータで学習でき，予測に適している．本提案の予測モデルを構築するキーアイディアは，構造方程式モデリングで得られた構造を用いて，ベイジアンネットワークで予測モデルを構築するというものである．

階層ベイズモデルは，統計モデルのパラメータに階層構造を持たせてベイズ推定するモデルであり，ベイジアンネットワークの一種である．将来のバグトピック発生数を一般化線形混合モデルで表現し，ファイルの個体差も含めた階層ベイズモデルを構築する [19]．まず，それぞれの将来バグトピック発生数 ATi をポアソン分布で表現する．これは，バグトピック数が離散値で，ゼロ以上の範囲となるからである．ポアソン分布は，平均 λ をパラメータとして持つ．このパラメータ λ を，構造方程式モデリングで得られた，説明変数となるメトリクスで表現する．潜在変数 Factor は，平均 μ と分散 σ のパラメータを持つ正規分布で表現した．パラメータ μ は計測したメトリクスでモデル化し，非負のパラメータ σ は無情報事前分布で与えた．無情報事前分布とは，$[-\infty, \infty]$ の範囲で任意の値をとってよ

い分布で，ここでは $[0, 100]$ の範囲の一様分布とした．

潜在変数を含む全てのパラメータは，マルコフ連鎖モンテカルロ（MCMC）サンプリングで推定する．これには，R の *Rstan* パッケージを用い，2,000 個 ×4 ステップのサンプル列を得た．将来バグトピック発生数 ATi について得られたサンプル列の中央値を予測値とする．

6. 評価実験

3.2 節で示した 2 つのプロジェクトそれぞれで，直前のバージョンを学習させ予測する．すなわち，バージョン 3.4 を学習しバージョン 3.5 の将来欠陥トピックを予測する実験と，バージョン 3.5 を学習しバージョン 3.6 の将来バグトピックを予測する実験を行う．

6.1 評価指標

予測モデルは将来のバグトピック発生数を算出する．ただし，発生数の精度ではなく，各ファイルに特定のバグトピックが将来発生することを正しく予測できたかという観点で評価する．評価指標には，Precision-Recall 曲線の作る面積，AUPRC（Area Under the Precision-Recall Curve）を用いる．AUPRC は，0 から 1 の範囲を取り，値が大きいほどよい予測モデルといえる．Precision と Recall はトレードオフの関係にあるので，一方を高くすると他方が低くなる．AUPRC が高いことは，両方が高いことを意味する．

正例（Positive）を正しく予測できたものを True Positive（TP），正例を負と予測したものを False Positive（FP），負例（Negative）を正しく予測できたものを True Negative（TN），負例を正と予測したものを False Negative（FN）と呼ぶ．

Recall は，全ての正例のうち正しく正と予測されたものの割合である．

$$\text{Recall} = \frac{TP}{TP + FN}$$

Precision は，正と予測されたもののうち実験に正であるものの割合である．

$$\text{Precision} = \frac{TP}{TP + FP}$$

6.2 結果

提案した予測モデルの有効性を評価するため，従来のバグ予測研究でよく用いられる線形回帰モデルと比較する．線形回帰モデルでは，全ての計測メトリクスを用いて各バグトピック発生数を予測するモデルを構築する．この回帰モデル構築には，R の glm() 関数を用いた．

表 3 は，プロジェクト JDT のバージョン 3.4 で学習し，バージョン 3.5 の将来バグトピックを予測した結果である．また，**表 4** は，プロジェクト PDE のバージョン 3.4 で学習し，バージョン 3.5 の将来バグトピックを予測した結果である．各表は，バグトピック毎に従来モデル（回帰モデル）と提案モデルの AUPRC を示す．また表の 2 列目は，各トピックのバグ有りファイルのパーセンテージである．従来モデルと提案モデルで比較し，高い AUPRC 値を太字で示す．バグ有りファイルのパーセンテージが低い（10% 未満）バグトピックの行の背景を灰色にした．

表 3 JDT の予測結果 AUPRC（バージョン 3.4 学習，バージョン 3.5 予測）

Table 3 AUPRC of the JDT Results (Training: 3.4, Prediction: 3.5).

トピック	バグ有り%	従来	提案
1	11	0.41	**0.56**
2	6.6	0.23	**0.27**
3	13	**0.58**	0.56
4	4.7	**0.50**	0.45
5	8.8	0.13	**0.25**
6	3.4	0.12	**0.23**
7	1.8	0.04	**0.09**
8	5.6	0.28	**0.32**
9	11	0.56	**0.57**
10	18	0.42	**0.61**

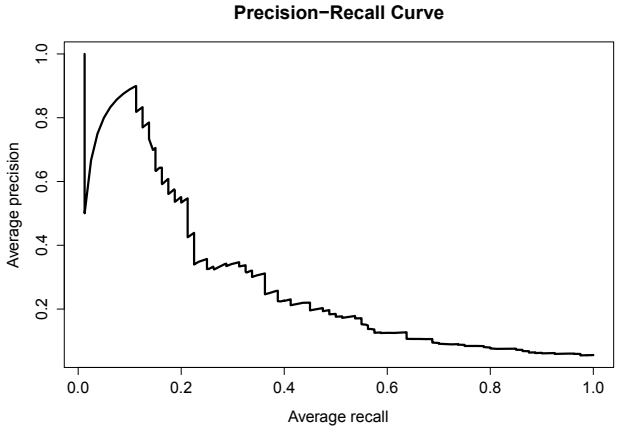
表 4 PDE の予測結果 AUPRC（バージョン 3.4 学習，バージョン 3.5 予測）

Table 4 AUPRC of the PDE Results (Training: 3.4, Prediction: 3.5).

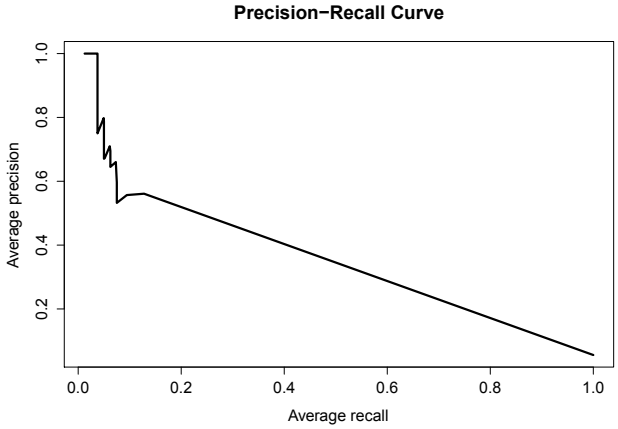
トピック	バグ有り%	従来	提案
1	5.0	**0.35**	0.32
2	4.2	0.16	**0.17**
3	7.1	**0.44**	0.37
4	2.6	0.160	**0.161**
5	0.14	0.005	**0.007**
6	3.0	0.22	**0.23**
7	0.36	0.004	**0.019**
8	6.8	**0.35**	0.34
9	8.8	0.20	**0.24**
10	27	0.499	**0.501**

図 4 は，プロジェクト JDT における従来モデルと提案モデルの予測結果の Precision-recall 曲線である．バグ有りファイルのパーセンテージが低い場合，従来モデルは図 4 のように，Recall が高い場合に Precision が低くなりやすい [10,13]．一方，同じ状況で提案モデルは，Recall が高い場合も Precision が高く，高い AUPRC となっている．ただし，バグ有りファイルの割合が特に低い場合（1% 未満，濃い灰色の背景），提案モデルでも従来モデル同様うまく予測できなかった．また，バグ有りファイルの割合が高い場合（10% 以上，白色の背景），従来モデルでもよく予測できているため，著しい改善は無かった．

プロジェクト JDT は 10 中 8 つのバグトピックで提案モデルが高い値を得た．プロジェクト PDE では 10 中 7 つの



(a) 従来モデルの結果（JDT バグトピック 8）



(b) 提案モデルの結果（JDT バグトピック 8）

図 4 Precision-recall 曲線
Fig. 4 Precision-recall curve.

バグトピックで提案モデルが高い値を得た．プロジェクト JDT と PDE のバージョン 3.5 で学習し，バージョン 3.6 を予測した結果でも，多くのバグトピックで改善が見られる一方，一部のバグトピックではよい結果が得られなかった．この結果を次の節で議論する．

7. 議論

7.1 妥当性への脅威

本稿では，バグの分類にトピックモデリングを適用した．しかし得られたバグトピックはその解釈が容易ではない．今後，より洗練されたバグ分類技術の開発によって，より有用なバグの識別ができることを期待する．一方，提案した予測モデルは，バグトピックに限定しない．あらかじめ分類されたバグ情報があれば，バグトピックと同様にモデルを構築できると考えている．ただし，分類方法と予測精度について明らかではないので実証的な分析が必要である．

対象プロジェクトは Eclipse から選択した 2 つのオープンソースソフトウェアである．そのため本結果の一般性へは妥当性の脅威がある．今後の課題として，多様なプロジェクトを分析対象とすることが挙げられる．

表 5 PDE 結果詳細（バージョン 3.5 学習，バージョン 3.6 予測）
Table 5 Details of PDE Prediction Results (Training: 3.5, Prediction: 3.6).

Topic	バグ有り%	AUPRC	誤差分散 3.5	誤差分散 3.6
1	3.4	0.18	79	87
2	1.2	**0.08**	83	90
3	3.9	0.17	79	69
4	2.1	0.14	57	86
6	1.6	**0.11**	96	98
8	3.8	0.22	41	84
9	5.2	**0.10**	67	74

7.2 予測精度が高くない結果の分析

表 5 に，プロジェクト PDE のバージョン 3.5 を提案モデルで学習し，バージョン 3.6 の将来バグトピックを予測した結果とモデルの情報を載せる．特に，バグ有りファイルの割合が小さい場合の結果を分析するため，バグ有りファイルのパーセンテージが 10% 未満のバグトピックのみを対象とする．また，1% 未満の極端なものも除去した．

従来モデルの結果と比べて高い値となった AUPRC を太字で示している．この予測結果はあまりよくない．この結果をモデルから分析する．図 3(b) が予測モデル構築時に利用した構造方程式モデリングの結果である．この潜在変数の誤差分散が 0.65 と高い．対象の 7 トピック全てがこの潜在変数に依存するため，AUPRC が低くなったと考えられる．7 トピック全体に AUPRC は低いが，トピック 8 は他よりやや高い．この将来バグトピック発生数メトリクス AT8 の誤差分散は，他より低い（44）．そのため相対的にうまく予測できたと考えられる．

7.3 予測精度の改善案

メトリクスの追加

潜在変数の誤差分散値が高いモデルは，収集したメトリクスでは十分潜在変数を説明できなかったと解釈できる．有用なメトリクスを追加し，潜在変数の誤差分散値を下げることは予測精度向上に有効と考えられる．追加するメトリクスは，通常のバグ予測にこれまで提案されてきたメトリクスから検討するとともに，新規にメトリクスを開発することも試みる．

時系列変化を考慮したモデル構築

提案モデルは，異なるバージョン間で得られる構造方程式モデリングのパス図が似た構造となることを利用している．本稿では，1 つ前のバージョンの構造を学習し，直後のバージョンで予測を行った．しかし詳細を分析すると，トピックによっては異なる構造となることがある．この時系列変化を考慮して予測モデルを構築できれば，高い予測精度を得られる可能性がある．

8. おわりに

本稿では，バグを一律に扱う従来のバグ予測モデルを拡張し，バグをトピック別に予測するモデルを構築した．予測対象バグの詳細化は，学習データ中の正例の割合が小さくなるため，予測精度が悪くなることが報告されている．我々は，構造方程式モデリングで変数間の因果関係や潜在構造を明らかにし，階層ベイズモデルでポアソン誤差を仮定した予測モデルを構築する手法を提案した．オープンソースソフトウェアを対象に，回帰モデルと比較実験を行った結果，高い AUPRC 値が得られることを確認した．また，構造方程式モデリングで得られたパス図の検討から，予測モデル改善案として，メトリクスの追加と時系列変化の考慮が挙げられる．

参考文献

[1] 畑　秀明，水野　修，菊野　亨：不具合予測に関するメトリクスについての研究論文の系統的レビュー，コンピュータソフトウェア，Vol. 29, No. 1, pp. 106–117 (2012).

[2] Chidamber, S. R. and Kemerer, C. F.: A Metrics Suite for Object Oriented Design, *IEEE Trans. Softw. Eng.*, Vol. 20, pp. 476–493 (1994).

[3] Subramanyam, R. and Krishnan, M. S.: Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects, *IEEE Trans. Softw. Eng.*, Vol. 29, pp. 297–310 (2003).

[4] Nagappan, N. and Ball, T.: Use of relative code churn measures to predict system defect density, *Proc. of 27th Int. Conf. on Softw. Eng.*, ICSE '05, pp. 284–292 (2005).

[5] Graves, T. L., Karr, A. F., Marron, J. S. and Siy, H.: Predicting Fault Incidence Using Software Change History, *IEEE Trans. Softw. Eng.*, Vol. 26, pp. 653–661 (2000).

[6] Hassan, A. E.: Predicting faults using the complexity of code changes, *Proc. of 31st Int. Conf. on Softw. Eng.*, ICSE '09, Washington, DC, USA, IEEE Computer Society, pp. 78–88 (2009).

[7] Nagappan, N., Murphy, B. and Basili, V.: The influence of organizational structure on software quality: an empirical case study, *Proc. of 30th Int. Conf. on Softw. Eng.*, ICSE '08, pp. 521–530 (2008).

[8] Bird, C., Nagappan, N., Devanbu, P., Gall, H. and Murphy, B.: Does distributed development affect software quality? An empirical case study of Windows Vista, *Proc. of 31st Int. Conf. on Softw. Eng.*, ICSE '09, pp. 518–528 (2009).

[9] Bird, C., Nagappan, N., Murphy, B., Gall, H. and Devanbu, P.: Don't touch my code!: examining the effects of ownership on software quality, *Proc. of 8th Joint Meeting of the European Softw. Eng. Conf. and the ACM SIGSOFT Symp. on the Found. of Softw. Eng.*, ESEC/FSE '11, pp. 4–14 (2011).

[10] Shihab, E., Mockus, A., Kamei, Y., Adams, B. and Hassan, A. E.: High-impact defects: a study of breakage and surprise defects, *Proc. of 8th Joint Meeting of the European Softw. Eng. Conf. and the ACM SIGSOFT Symp. on the Found. of Softw. Eng.*, ESEC/FSE '11, pp. 300–310 (2011).

[11] Shin, Y., Meneely, A., Williams, L. and Osborne, J. A.: Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities, *IEEE Trans. Softw. Eng.*, Vol. 37, No. 6, pp. 772–787 (2011).

[12] Zimmermann, T., Nagappan, N. and Williams, L.: Searching for a Needle in a Haystack: Predicting Security Vulnerabilities for Windows Vista, *Proc. of 3rd Int. Conf. on Softw. Testing, Verification and Validation*, ICST '10, pp. 421–428 (2010).

[13] Menzies, T., Dekhtyar, A., Distefano, J. and Greenwald, J.: Problems with Precision: A Response to "Comments on 'Data Mining Static Code Attributes to Learn Defect Predictors'", *IEEE Trans. Softw. Eng.*, Vol. 33, No. 9, pp. 637–640 (2007).

[14] Lukins, S. K., Kraft, N. A. and Etzkorn, L. H.: Bug localization using latent Dirichlet allocation, *Inf. Softw. Technol.*, Vol. 52, No. 9, pp. 972–990 (2010).

[15] Nguyen, A. T., Nguyen, T. T., Al-Kofahi, J., Nguyen, H. V. and Nguyen, T. N.: A topic-based approach for narrowing the search space of buggy files from a bug report, *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, ASE '11, pp. 263–272 (2011).

[16] Nguyen, A. T., Nguyen, T. T., Nguyen, T. N., Lo, D. and Sun, C.: Duplicate bug report detection with a combination of information retrieval and topic modeling, *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, ASE '12, pp. 70–79 (2012).

[17] Thomas, S. W.: Mining software repositories using topic models, *Proc. of 33rd Int. Conf. on Softw. Eng.*, ICSE '11, pp. 1138–1139 (2011).

[18] Kim, S., Zimmermann, T., Whitehead Jr., E. J. and Zeller, A.: Predicting Faults from Cached History, *Proc. of 29th Int. Conf. on Softw. Eng.*, ICSE '07, pp. 489–498 (2007).

[19] 久保拓弥：データ解析のための統計モデリング入門: 一般化線形モデル・階層ベイズモデル・MCMC，岩波書店 (2012).

[20] 狩野　裕：構造方程式モデリングは, 因子分析, 分散分析, パス解析のすべてにとって代わるのか?< 特集 > 討論:共分散構造分析)，行動計量学，Vol. 29, No. 2, pp. 138–159 (2002).

[21] Anderson, R. D. and Vastag, G.: Causal modeling alternatives in operations research: Overview and application, *European J. of Operational Research*, Vol. 156, No. 1, pp. 92 – 109 (2004).

[22] Gupta, S. and Kim, H. W.: Linking structural equation modeling to Bayesian networks: Decision support for customer retention in virtual communities, *European J. of Operational Research*, Vol. 190, No. 3, pp. 818 – 833 (2008).