6A-05

ナイブベイズを用いた ソフトウェア不具合の修正者自動割り当て手法

田中 裕大† 加瀬 直樹† 夏目 珠規子† 市田 憲明†

株式会社 東芝†

1. はじめに

近年，ソフトウェアの大規模化や複雑化が進んできており、それに伴いソフトウェアの不具合の修正にかかるコストも増加している．多くの組織・プロジェクトにおいて，ソフトウェアの不具合を修正する際は，該当不具合の修正を，適切な修正者への割り当てる必要がある．しかし，開発者の増加やソフトウェア自体の複雑化により，適切な修正者へ割り当てることが難しくなってきている．本手法では，不具合管理システムに登録されている不具合カテゴリ情報から，ナイブベイズを用いて割り当てる修正者を予測・推薦する．また，実プロジェクトのデータを用いて，振分者が人手で修正者へ割り当てる場合と，本手法により自動で修正者を割り当てる場合を比較した結果について述べる．

2. 不具合修正における課題

発見された不具合を修正するプロセスは組織やプロジェクトによって様々であるが，一般的には以下のようなフローをたどることが多い．

(1) 振分者が不具合の修正者を割り当てる
(2) 修正者が不具合を修正する
(3) 発見者が不具合の修正を確認する

ソフトウェア規模が小さい場合，(1)の修正者の割り当てにおいて適切な修正者へ割り当てることは難しくない．しかし，ソフトウェアの規模が大きくなるに伴い，ソフトウェア自体の複雑化や開発者の増加により，ソフトウェア不具合の適切な修正者への割り当てが難しくなってくる．適切でない修正者に割り当てた場合，その修正者が不具合現象の確認やデバッグを行うコスト増加や不具合の修正が完了するまでのターンアラウンドタイムの増加をまねく． 社内のあるプロジェクトでは，システムテストにおいて，約 47%の不具合が最初に割り当てられた修正者と実際の修正者が異なっているという結果が出た．これらの割り当てミスによる後戻り工数は，不具合修正プロセス全体の約 24%を占めていた．

Automatic method to assign the person in charge of fixing software bugs with Naive Bayes classifier.
† Yuta Tanaka, Naoki Kase, Mikiko Natsume, Noriaki Ichida, Toshiba Corporation Industrial ICT Solutions Company

3. 不具合管理システムを活用した修正者割り当て自動化手法

多くの組織・プロジェクトにおいて，不具合が適切に修正されることを管理するために不具合管理システムが活用されている．不具合管理システムは，「不具合の現象」「発見したテスト番号」「発生頻度」「修正確認結果」など不具合の属性を表すフィールドを定義する機能や，不具合修正のワークフロー管理の機能を持っていることが多い．東芝でも，不具合管理システム PRISMY を開発しており，社内・社外の多くのユーザに利用されている．

不具合管理システムには発見された不具合に関する多くの情報が記録されているが，従来，これらの情報の多くはその不具合を修正することのみに利用され，修正完了後には利用されないことが多かった．しかし，ソフトウェアリポジトリマイニング技術の発展により，不具合管理システムのデータから有用な情報を抽出する研究が行われている．[1][2]

本手法では，過去に発見された不具合の情報を用いて，新規に発見された不具合の修正者を推定する．不具合管理システムには，該当プロジェクトの過去に発見された不具合に関する情報が多く保存されている．不具合の情報は不具合ランクなど，いくつかの選択肢から 1 つもしくは複数を選択する「カテゴリ情報」と不具合の現象など，自然言語で記述する「自由記述情報」の 2 つに大別できる．本手法では，推定元となるデータにカテゴリ情報を用いる．例えば，表 1の「発見者」「試験カテゴリ」「発生 HW」などがカテゴリ情報に当たる．

表 1 過去の不具合の例

ID	発見者	試験カテゴリ	発生 HW	修正者
#1	A さん	機能	HW-A	F さん
#2	A さん	セキュリティ	HW-B	D さん
#3	B さん	性能	HW-D	F さん
#4	C さん	機能	HW-C	D さん
#5	C さん	性能	HW-A	E さん

表 1 のような過去の不具合情報がある状況で新規に発見された不具合(表 2)の修正者を推測する．具体的には，各カテゴリ(「発見者」「試験カテゴリ」「発生 HW」など)を特徴変数としたナイブベイズを用いる．ナイブベイズは確率に基づいた教師あり分類器である．推定元となる変数の独立性を仮定することで，少ない学習データに適用でき高速に動作するという特徴を持つ．なお，今回はラプラススムージングを用いて，ナイブベイズのゼロ頻度問題を回避した．

表 2 新規不具合の例

ID	発見者	試験カテゴリ	発生 HW	修正者
#6	A さん	性能	HW-A	

上記例において，修正候補者が#6 を修正する確率を求めたところ，表 3のようになる．よってこの例では，推定結果として F さんを提示する．なお，ナイブベイズの変数の独立性の仮定，およびラプラススムージングにより，確率の合計値は 100%とならないことがある．

表 3 不具合#6 の推定結果

修正候補者	推定結果(%)
D さん	10.0
E さん	19.5
F さん	40.0

4. 検証事例

本手法を，すでに開発が完了している 1 件の実プロジェクトの不具合データに対し適用し，効果を測定した．

このプロジェクトでは，1183 件の不具合が検出された．推定に用いたカテゴリ(特徴変数)の数は 28 個であり，1 件以上の不具合を修正した修正者数は 76 人である．これを，以下の 2 種類の方法で推定し，推定結果の正解率を比較した．

(A) 振分者が人手で修正者を割り当てる場合
本プロジェクトは実プロジェクトであるため，不具合管理システムのワークフロー履歴を用いて算出した．振分者が修正者を割り当てた回数が 1 度のみの場合を正解，2 度以上の場合を不正解としている．

(B) 本手法により修正者を自動で割り当てた場合
プロジェクトの不具合管理システムのデータをもとに，クロスバリデーション(k=12)を行い算出した．

(A)と(B)を比較した結果は，表 4のようになった．

表 4 修正者割り当て正解率の比較

	(A)	(B)
正解率	約 53%	約 74%

(A)の人手による割り当てより，(B)の本手法による動割り当てのほうが，正解率が 21 ポイント高い結果となった．

正者割り当ての失敗率が 47%から 26%へ約 45%削減されており，割り当ての失敗による後戻りが不具合修正プロセス全体の約 24%を占めていることと合わせると，不具合修正時間が約 11%削減できると試算できる．また，不具合振分の自動化により，振分者が不具合の修正者への振分時間も削減できることが期待できる．

5. まとめ

本論文では不具合管理システムに登録された過去の不具合データを用いることで，新規に発見された不具合の修正者を予測する手法を提案した．また，実際のプロジェクトデータを用いて本手法を検証した結果，人手による割り当てより高い正確度で修正者を予測することができた．これにより，割り当てミスによる後戻りコストや振分の自動化による振分コストの削減が期待できる．

今後の課題として，自由記述情報の活用が挙げられる．文献[2]では，自由記述情報を用いて修正者を推定している．本手法では，多くのカテゴリ情報を活用することで高い正確度を実現しているが，「不具合の現象」などの自由記述情報には，カテゴリ情報には入っていない不具合の情報が記述されているため，この情報を活用することでさらに高い正確度で推定できる可能性がある．そこで，現在自由記述情報の活用を進めているが，文献[2]と同様の自然言語処理(文章の長さで正規化した単語の出現頻度算出)を行い検証事例で述べたプロジェクトデータに適用したところ，正確度の向上は見られなかった．今後は，自由記述情報から，新規不具合の修正者推薦に利用できる情報を抽出する手法を検討したい．

参考文献

[1] Per Runeson, Magnus Alexandersson and Oskar Nyholm; Detection of Duplicate Defect Reports Using Natural Language Processing; ICSE'07 pp499-510; 2007

[2] John Anvik, Lyndon Hiew, Gail C. Murphy; Who should fix this bug?; ICSE'06; pp361-370; 2006