6A-2

データマイニング手法を用いたプロダクトメトリクスの類似性に基づくソフトウェアインスペクション方法の提案

西川 雅彦† 青山 幹雄‡

南山大学 大学院 数理情報研究科† 南山大学 情報理工学部 ソフトウェア工学科‡

1. はじめに

ソフトウェアの再利用によって開発が進行すると再利用元で発見できなかったバグが再利用先に継承される．再利用元と再利用先のプロダクトメトリクス類似性に注目し，データマイニング手法を用いたインスペクション方法を提案する．

2 関連研究

(1)ソフトウェア欠陥の有無を推定する研究がある[1].
(2)インスペクションの工数最適化の研究がある[2].

3. アプローチ

データマイニング手法を用いてプロダクトの類似度でバグ発生を予測する．仮説検証として，仮説 1 は，再利用ソフトウェアとの類似度，仮説 2 では，再利用元と再利用先とのプロセスの類似度を明らかにする．

4. データマイニングに基づく分析方法の提案

データベース化された再利用元と再利用先のメトリクスをデータマイニング手法によって類似度比較し，プロセス類似バグと構造類似バグに分類することでインスペクション精度の向上策を提案する(図 1)．

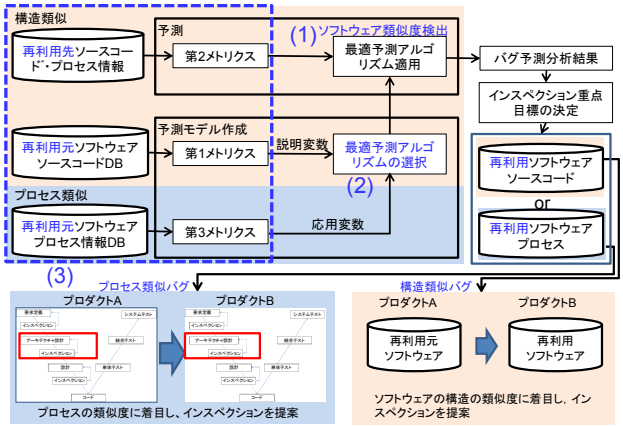

図 1 データマイニングに基づく分析方法の提案

(1)ソフトウェアの類似度の検出

仮説 1 構造類似バグに基づき，バグが発生したメトリクスの類似度に着目し，Weka [3]を利用して潜在する再利用先のバグを明らかにする．ソフトウェアの類似度予測値を評価し，Rating 比較することでソフトウェアの類似度を検証する．

(2)最適アルゴリズムの選択

Weka に実装されている複数の予測アルゴリズムを数種類選択し，再利用元と再利用先の類似度の予測値を評価し，Rating を比較することで最も精度が高いアルゴリズムを選択する．

(3)メトリクスデータ管理

再利用元と再利用先のバグ発生原因のプロセス情報メトリクスをデータベースで管理し，Weka の木構造解析を行うことで，再利用元と再利用先について統計的手法を用いて比較，検証する．

5. 実データへの提案方法の適用

5. 1. データセット

ソフトウェアのメトリクスとバグデータとして Promise Software Engineering Repository[4] を利用する対象は，宇宙機器(CM1, KC1)，科学データ処理(KC2)，機載ソフトウェアである．

5. 2. 仮説 1 構造類似バグの検証

再利用元と再利用先を比較し，予測アルゴリズムによってメトリクスの類似度を評価する．

(1)検証方法

複数あるソフトウェアの中から，複雑度メトリクスの予測値が最も近いソフトウェアを同レベルのプロダクトデータとして再利用元と再利用先の予測値を求める．

複雑度メトリクスによる予測値が，

再利用元予測値＞再利用先予測値

の関係であれば，再利用元ソフトウェアを仕様変更したことでソフトウェア構造が複雑となったことを表す．

ソフトウェアモジュール中にバグの発生割合が，

再利用元 ＜ 再利用先

の関係であれば，再利用先でバグが引き継がれていると定義する．

(2)最適アルゴリズムの選択

最適アルゴリズムの選択方法としては，Weka の類似度予測結果の各項目の精度に Rating を付与し，平均が最大値 10 に近ければ最適アルゴリズムとなる．

Weka で実装されている各カテゴリで代表的なアルゴリズムを比較したところ，MultilayerPerceptron が平均 10 であり，最も予測値が高いことが分かった．

(3)類似プロダクトデータの選出

最適アルゴリズム MultilayerPerceptron を使用して複雑度メトリクスの予測値を算出し，得られた予測値をもとに Rating を付与したところ，同レベルの類似度のソフトウェアが CM1, KC1, KC2 であった．この 3

A Software Inspection Method Based on the Structural Similarity of the Components Using Data Mining Method
†Masahiko Nishikawa, Graduate School of Mathematical Sciences and Information Engineering Nanzan University
‡Mikio Aoyama, Dep. of Software Engineering Nanzan University

つの類似プロダクトデータを再利用元と再利用先データの定義候補のソフトウェアとした.

(4) 再利用先と再利用元ソフトウェアの定義決め

ソフトウェアを再利用すると, ソフトウェア構造が複雑化して各データ間の類似度が低くなる. したがって複雑度を表すメトリクスとバグ発生の有無に限定し, Weka の予測結果を基に類似度に対して Rating を付与した. Rating 3.0 に近ければ、複雑度が低く, 仕様変更前の再利用元ソフトウェアと定義できる(表 1).

再利用元 Rating ＞ 再利用先 Rating

の関係が成り立てば, 類似度の高低によって仕様変更前後の関係となる.

表 1 仕様変更前後のソフトウェアの定義結果

再利用元ソフトウェア	Rating Avg.	再利用先ソフトウェア	Rating Avg.
CM1	2.5	KC1	2.0
CM1	2.5	KC2	1.5
KC1	2.0	KC2	1.5

(5)再利用先から再利用元へのバグ引継確認結果

各ソフトウェアモジュール単位のバグ発生割合を求めた(表 2). 次に①再利用元のバグの発生割合と②再利用先のバグ発生割合を比較すると①＜②の関係が成立し, 類似度比較によってバグ引継が確認できることを明らかにした(表 3).

表 2 各ソフトウェアモジュール中のバグ発生割合

Category	①バグなしモジュール数	②バグありモジュール数	②/①*100
CM1	449	49	10.9 %
KC1	1783	326	18.3 %
KC2	415	107	25.8 %

表 3 再利用先から再利用元へのバグ引継確認結果

①再利用元ソフトウェア	バグの発生割合	②再利用ソフトウェア	バグの発生割合	①②の関係性
CM1	10.9	KC1	18.3	①<②
CM1	10.9	KC2	25.8	①<②
KC1	18.3	KC2	25.8	①<②

5. 3. 仮説 2 プロセス類似バグの検証

バグの原因を①要求定義のエラー, ②コードエラー, ③仕様書のエラーのいずれかのエラーであるか Weka の木構造によって検証する.

(1)検証方法

再利用先でバグが発生したデータを①～③に分類し、データベース化した.

(2)最適アルゴリズムの選出

Weka で実装されている Trees のカテゴリを代表するアルゴリズムで分類精度の確認を行ったところ, J48-C 0.25-M2(以下 J48)が最も高いことが分かった.

Trees 以外のアルゴリズムには, J48 よりも高い分類精度を持つものがある. 予測値比較すると平均 0.963 であり, 有意水準 95%以内で信頼区間を得ているため J48 の予測アルゴリズムは妥当性を得る.

6. 評価結果

(1)バグ原因の層別化

図 2 に示す通り, Weka に実装されている木構造の視覚化により, バグ発生の要因を解析する. 再利用ソフトウェアで発生したバグの中から発生原因を①要求定義のバグ, ②コードバグ, ③仕様書バグに層別し再利用元プロセスの問題点を明らかにする.

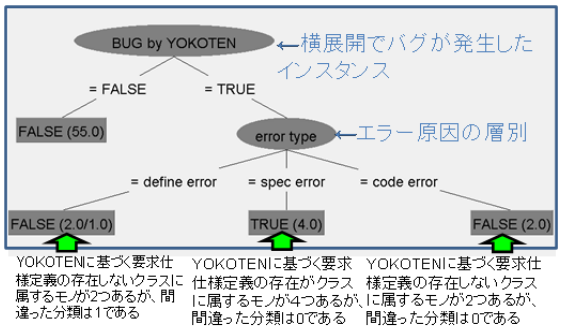

図 2 木構造のビジュアル結果

(2)バグ発生箇所の読み取り

表 4 に示す通り, 再利用先バグの原因の中に, ソフトウェア要求仕様定義に基づいたプロセスが存在しているときの分類精度を表している.

①より, 要求定義のバグは, クラスに属する総数に対して, 50%正しい結果が得られていないので除外.

②より, コードのバグは, 再利用元に要求仕様の定義が無くても, 再利用プログラムのバグを引き継ぐ.

③より, 仕様書のバグは, 再利用元に要求定義があり, 要求定義と仕様書に乖離がある. 仮説 2 のプロセスの類似性については①, ②, ③のいずれかが再利用元プロセスと同じであれば, バグの発生を回避することができない. これにより, ソフトウェアの複雑性に起因するバグが発生している可能性が明らかとなった.

これにより, バグが発生した再使用元と再使用ソフトとの類似性が高ければバグ発生の可能性が高まり, プロセスに課題があれば, 再利用時にバグを引き継がれることが明らかとなった.

表 4 バグ発生原因と要求仕様プロセス有無の分類

	バグ原因	YOKOTEN制御に基づいた要求仕様の有無	クラスに属するインスタンス(A)	間違った分類(B)	(B)/(A)%
①	要求定義のバグ(define error)	×	2	1	50
②	コードのバグ(code error)	×	2	0	100
③	仕様書のバグ(spec error)	○	4	0	100

7. まとめ

統計的観点からデータマイニングツールを活用し, 簡易な分析によってソフトウェア開発手法とソフトウェアコードの類似度を比較し, バグが発生する要因を特定, 予測を行うことでソフトウェアインスペクション精度を向上する手法を提案した.

参考文献

[1] R. Subramanyan, et al., Empirical Analysis of CK Metrics for Object-Oriented Design Complexity, IEEE TOSE., Vol. 29, No. 4, 2003, pp. 297-310.

[2] 阿萬 祐久, 山下 裕也, 整数計画法を用いた重点レビュー対象モジュールの選択, コンピュータソフトウェア, Vol. 27, No. 4, 2010, pp. 240-245.

[3] Weka: Univ. of Waikato, http://www.cs.waikato.ac.nz/ml/weka/.

[4] J. S. Shirabad, et al., The PROMISE Repository of Software Engineering Databases, Univ. of Ottawa, http://promise.site.uottawa.ca/SERepository/.