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

M2012MM028　西川　雅彦

指導教員　青山　幹雄

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 Fault 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. 研究課題

個人スキルに依存したインスペクションではなく，データマイニングツールを活用した数学的な分析によって Fault が発生する要因を特定，予測を行うことで精度の高いソフトウェアインスペクションを提案する．

3. 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].

4. 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 Fault 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．

5. 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)．

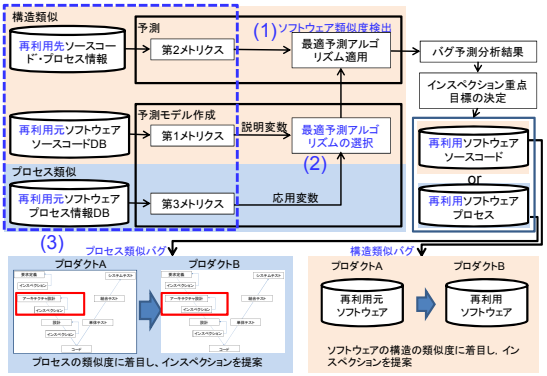


図 1 データマイニングに基づく分析方法の提案

(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，Fault 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 Fault 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(表 1).

表 1　最適アルゴリズムの選択方法

予想結果	内容
Precision（適合率）	Trueと分類された中で、実際にTrueであるモジュールの割合
Recall（再現率）	全体のTrueモジュールの中で正しくTrueと分類された割合
F-measure（F値）	適合率と再現率の調和平均
ROC area	ROC曲線下の面積は分類アルゴリズムの精度の高さを表す

予測結果の項目ごとの精度にRatigを付与し，その平均が最大値10に近いアルゴリズムが最適アルゴリズムとなる．

(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(図 2)．

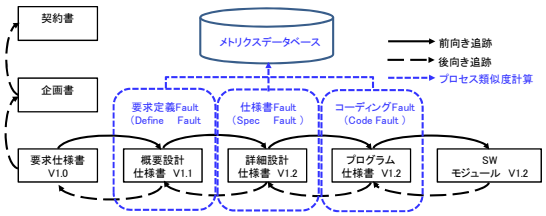


図 2 要求追跡方向と Fault 発生エラーの関係図

6. 実データへの提案方法の適用

6. 1. データセット

(1)プロダクトの選択

ソフトウェアのメトリクスと Fault 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．

(2)メトリクスの選択

Promise で公開されているプロダクトのメトリクスは 22 種類用意されているが，プロダクトに共通するメトリクスのみを選択して使用する．本研究で使用したメトリクスは主としてソース容量，ソフトの構成の複雑度，工数と個別にソフトウ

ェアの性格を表す個別案件で分類できるようにした(表 2).

表 2　複雑度メトリクスの種類と内容

メトリクスの種類	内容
ソース容量	McCabe's line count of code
	Halstead ″volume″
	Halstead ″program length″
複雑度	McCabe ″cyclomatic complexity″
	McCabe ″essential complexity″
	Halstead ″difficulty″
	Halstead ″intelligence″
工数	Halstead ″effort″
その他	Halstead's count of lines of comments
	IOCodeAndComment: numeric
	total operators
	total operands
	of the flow graph

6. 2. 仮説 1 構造類似性バグの検証

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

表 1 のデータを，Weka を使用して予測精度が最も近いデータを類似プロダクトデータとし，再利用元ソフトウェアと再利用ソフトウェアと定義する.

表 2 で示す複雑度のメトリクスによる予測精度が，

再利用元予測精度 ＜ 再利用予測精度

の関係であれば，再利用元ソフトウェアに基づいて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 Fault の発生割合が，

再利用元ソフトウェア ＜ 再利用ソフトウェア

の関係であれば，再利用ソフトウェアで Fault が引き継がれていることを表す.

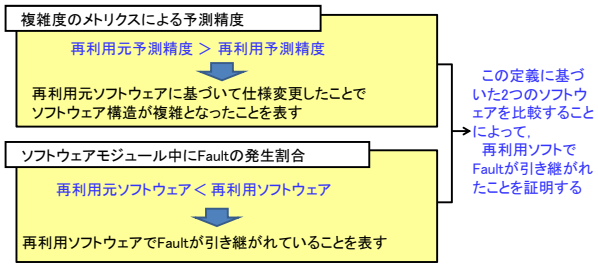


図 2　構造類似性バグの検証方法概念

(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 の予測結果の項目ごとの精度に Ratig を付与し，その平均が最大値 10 に近いアルゴリズムが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抽出し，Precision(適合率)，Recall(再現率)，F-measure(F 値), ROC area の観点で Rating 付与した他のアルゴリズムと比較したところ，MultilayerPerceptron が最も予測精度が高いことが分かった(表 3).

表 3　アルゴリズムの予測精度評価結果

Category	Name	Precision	Rating	Recall	Rating	F-Measure	Rating	ROC Area	Rating	Rating Avg.
trees	J48	0.970	8	0.952	8	0.957	8	0.972	4	7.0
	J48graft	0.950	2	0.937	5	0.941	5	0.874	5	4.3
	RandomForest	0.928	3	0.937	5	0.929	2	0.966	3	3.3
	LADTree	0.900	1	0.921	1	0.905	1	0.369	1	1.0
functions	Logistic	0.937	4	0.937	5	0.937	3	0.983	7	4.8
	MultilayerPerceptron	0.987	10	0.984	10	0.985	10	1.000	10	10.0
bayes	BayesNet	0.977	9	0.968	9	0.971	9	0.997	9	9.0
	BayesSimple	0.950	5	0.937	5	0.941	5	0.986	8	5.8
meta	AttributeSelectedClassifie	0.957	7	0.952	8	0.954	7	0.964	2	6.0
rules	PART	0.957	7	0.952	8	0.954	7	0.983	7	7.3

・Precision（適合率）：Trueと分類された中で、実際にTrueであるモジュールの割合
・Recall（再現率）　：全体のTrueモジュールの中で正しくTrueと分類された割合
・F-measure（F値）　：適合率と再現率の調和平均
・ROC area　　：ROC曲線下の面積は分類アルゴリズムの精度の高さを表す

(3)類似プロダクトデータの選択

表 3 のソフトウェアに最適アルゴリズム MultilayerPerceptron の予測精度を算出した(図 3, 4)

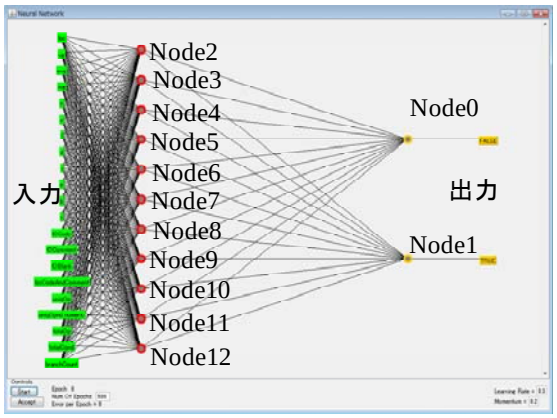


図3　WekaによるMultilayerPerceptronの入出力関係図

```
=== Summary ===
Correctly Classified Instances         436               87.5502 %
Incorrectly Classified Instances        62               12.4498 %
Kappa statistic                         -0.0156
Mean absolute error                      0.1568
Root mean squared error                  0.3121
Relative absolute error                 87.6482 %
Root relative squared error            104.785  %
Total Number of Instances              498
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.969     0.98      0.901     0.969     0.933      0.734    FALSE
                 0.02      0.031     0.067     0.02      0.031      0.734    TRUE
Weighted Avg.    0.876     0.886     0.819     0.876     0.845      0.734

=== Confusion Matrix ===
   a   b   <-- classified as
 435  14 |  a = FALSE
  48   1 |  b = TRUE
```

図 4　Weka による MultilayerPerceptron の計算結果

得られた予測精度をもとに表 4 に示す通り Rating を付与し，同レベルの類似性をもったソフトウェアを CM1, KC1, KC2 とした. JM1 と PC1 は Rating 結果が他の 3 つのソフトウェアと比較して大きく異なっているため，類似プロダクトデータとして選択ができないことが分かった(表 4).

表 4　類似プロダクトデータの選択結果

Category	Precision	Rating	Recall	Rating	F-Measure	Rating	ROC Area	Rating	Rating Avg.
CM1	0.819	2	0.876	4	0.845	4	0.734	3	3.3
JM1	0.769	1	0.810	1	0.741	1	0.690	1	1.0
KC1	0.835	4	0.859	3	0.828	2	0.771	4	3.3
KC2	0.834	3	0.847	2	0.835	3	0.828	5	3.3
PC1	0.921	5	0.936	5	0.917	5	0.723	2	4.3

(4)仕様変更前後のソフトウェアの選択

表 5 に示す通り，類似プロダクトデータの中で仕様変更後のソフトウェアを選択した.

再利用元ソフトウェアから再利用ソフトウェアを作成すると，ソフトウェア構造が複雑化して各データ間の類似性が低くなる. したがって，複雑度を表すメトリクスと Fault 発生の有無に限定したうえで，Weka の予測結果を基に Ratig を付与した.

全体のソフトウェア数が 3 であるため，複雑度が低くいソフトウェアを Rating 3.0 とし，仕様変更前でソフトウェアの複雑度が高くなる前の状態を定義する. 逆に複雑度が高いソフトウェアを Rating 1.0 とし，仕様変更後でソフトウェアの複雑度が高くなった状態を定義する.

再利用元類似性 Rating＞再利用ソフト類似性 Rating

の関係が成り立てば，類似性の高低によって仕様変更前後の関係となる(表 5).

表 5　仕様変更前後のソフトウェアの選択結果

Category	Precision	Rating	Recall	Rating	F-Measure	Rating	ROC Area	Rating	Rating Avg.
CM1	0.843	3.0	0.892	3.0	0.859	3.0	0.691	1.0	2.5
KC1	0.830	2.0	0.856	2.0	0.830	2.0	0.788	2.0	2.0
KC2	0.814	1.0	0.830	1.0	0.816	1.0	0.829	3.0	1.5

ただし，この時点では，CM1，KC1，KC2 のいずれのソフトウェアが再利用元あるいは，再利用先ソフトウェアであるのか不明であるため，CM1，KC1，KC2 のマトリクス比較によって ①式を満たす組み合わせを○で表記し，それ以外を×で表記する(表 6).

表 6 再利用元と再利用先複雑度 Rating マトリクス比較

	CM1 平均Rating 2.5	KC1 平均Rating 2.0	KC2 平均Rating 1.5
CM1 平均Rating 2.5		○	○
KC1 平均Rating 2.0	×		○
KC2 平均Rating 1.5	×	×	

仕様変更前の再利用元ソフトウェアと，仕様変更後の再利用先ソフトウェアを①式を満たす組み合わせをまとめると以下の 3 パターンの組み合わせとなる(表 7).

表 7 アルゴリズムの予測精度評価結果

仕様変更前		仕様変更後	
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

これによって，再利用元ソフトウェアと再利用先ソフトウェアの組み合わせの定義を決めることができた.

(5) 再利用ソフトウェアの Fault の引き継ぎ確認結果

複雑度のメトリクスの算出結果を基に定義した再利用元ソフトウェアと，再利用先ソフトウェアの Fault の発生割合が

再利用元の Fault 発生割合 ＜ 再利用先 Fault 発生割合

であれば再利用元ソフトウェアの Fault が引き継がれ，再利用先のソフトウェアの Fault の発生が増大したといえる. そこで，各ソフトウェアのモジュール中の Fault 発生割合を求めた(表 8).

表 8 各ソフトウェアモジュール中の Fault 発生割合

Category	①Faultなし モジュール数	②Faultあり モジュール数	②/①*100
CM1	449	49	10.9 %
KC1	1,783	326	18.3 %
KC2	415	107	25.8 %

次に表 6 に示した通り，再利用元ソフトウェアと再利用先ソフトウェアの組み合わせの定義結果に対して，表 7 の複雑度のメトリクスの算出結果を基に定義した再利用元ソフトウェアと，再利用先ソフトウェアの Fault の発生割合が同じであれば，以下の関係が成り立つ.

再利用元複雑度 Rating ＞ 再利用先複雑度 Rating

複雑度の高低によって再利用元（仕様変更前）と再利用先（仕様変更後）の関係が成立する（複雑度小＝Rating 大 ，複雑度大＝Rating 小）.

再利用元の Fault 発生割合 ＜ 再利用先 Fault 発生割合

再利用元ソフトウェアの Fault が引き継がれ，再利用先のソフトウェアの Fault の発生が増大したことを明らかにすることができる.

図 5 に示した通り，【A】再利用元ソフトウェアの Fault の発生割合，と【B】再利用先ソフトウェアの Fault の発生割合を比較すると全ての組み合わせで【A】＜【B】の関係が成り立ち，実データ上で Fault が引き継がれていることを明らかにすることができた.

これにより，再利用先のソフトウェアの Fault 発生を予測する手段として，予測値の最も高いアルゴリズムを用いて，モジュールの複雑度を算出し，Rating によって Fault 発生の高さを数値化することで Fault モジュールを特定することができる.

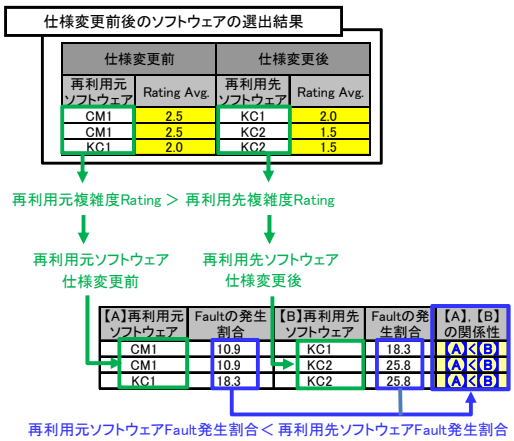


図 5 再利用ソフトウェアの Fault の引き継ぎ確認結果

6. 3. 仮説 2 プロセス類似性バグの検証

要求仕様書に記述されている要求項目が正当なものであるかどうかを確認するためには，仕様書の前提となった文書に当たって確認を取らなければならない. こうした作業を支援する手段として要求追跡がある.

要求を追跡するには，文書の作成時に，追跡リンクを設定しておく，要求追跡を実現するためには，それぞれの要求と，要求変更が公的な文書として文書化され，構成管理の下に管理されている必要がある.

再利用ソフトウェアにおける要求仕様書のプロセスへの期待は，熟練分析者の経験と知識を基にした属人的情報を記録し，その情報から，犯さなくとも済むはずの失敗を再び繰り返さなくてすむ可能性を高めることである.

要求仕様書には，ユーザーの要求が，出来るだけ正確かつ完全な形で表現されていなければならない，しかし，限られた時間内に，ユーザーが対象領域の知識と，システムに対する個別の要求のすべてを分析者に伝えることは不可能に近い. またそれらの要求を分析者が正しく理解し，正確に記述することも困難である. そのような場合，類似した問題領域での分析者の過去の経験が，ユーザーの知識の欠如と，分析者の理解不足を補うことができる.

そのため要求追跡のデータ管理方法として ①要求定義のエラー，②コードエラー，③仕様書のエラーの分類を定義し，Fault 発生原因を特定することで得られる検証結果をメトリクス化することで，プロセス類似度に Fault 発生の予測値を求めることを可能とする.

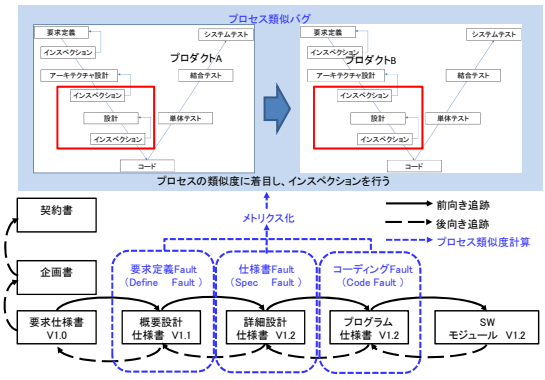


図 6 要求追跡の方向とプロセス類似バグ計算図

再利用データとそれ以外のデータを分類し，Fault 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

表 9 に示す通り，再利用ソフトウェアで Fault が発生したデータを①～③に分類した．Weka で実装されている Trees カテゴリの中から代表的なアルゴリズム 4 個のうち最も精度が高いものを選択する.

また，木構造によって定量的評価を行うため，それ以外のカテゴリのアルゴリズムと比較して，Trees のアルゴリズムで得られる予測精度に妥当性があるか検証する.

表 9 再利用バグの原因の一覧

Base No.	Development phase	Fountainhead of Demand	Details	BUG by "YOKOTEN"	Man day	Fault type	Existence of a requirement specification definition based on "YOKOTEN"
2	A	new		FALSE	10	Define Fault	FALSE
7	B	new		FALSE	15	Define Fault	FALSE
34	B	new old	Improvement in an actuator operation	FALSE	15	Spec Fault	FALSE
11	B	new old	Improvement in an actuator operation	FALSE	15	Spec Fault	FALSE
4	A	new old	FALSE	FALSE	10	Spec Fault	FALSE
10	B	new old	FALSE	FALSE	15	Spec Fault	FALSE
33	B	new old	FALSE	FALSE	15	Spec Fault	FALSE
5	B	sinki morikomi	FALSE	FALSE	15	Define Fault	FALSE
19	B	sinki morikomi	FALSE	FALSE	15	Define Fault	FALSE
20	B	sinki morikomi	FALSE	FALSE	15	Spec Fault	FALSE
39	C	siyou syusei	Improvement in an actuator operation	FALSE	10	Define Fault	FALSE
43	C	siyou syusei	Improvement in an actuator operation	FALSE	10	Define Fault	FALSE
45	C	siyou syusei	Improvement in an actuator operation	FALSE	10	Spec Fault	FALSE
47	C	siyou syusei	Improvement in an actuator operation	FALSE	10	Spec Fault	FALSE
49	C	siyou syusei	Improvement in an actuator operation	FALSE	10	Define Fault	FALSE
52	C	siyou syusei	Improvement in an actuator operation	FALSE	10	Spec Fault	FALSE
61	E	siyou syusei	Improvement in an actuator operation	FALSE	10	Spec Fault	FALSE
15	C	siyou syusei	Change state correction	TRUE	10	Define Fault	FALSE
27	E	siyou syusei	Change st	FALSE	10	Spec Fault	FALSE
28	E	siyou syusei	Change st		10	Define Fault	FALSE
37	C	siyou syusei	Change st		10	Spec Fault	FALSE
42	C	siyou syusei	Change st		10	Define Fault	FALSE
46	C	siyou syusei	Change st		10	Define Fault	FALSE
38	C	siyou syusei	Omission of specification	FALSE	10	Define Fault	FALSE
40	C	siyou syusei	Omission of specification	TRUE	10	Spec Fault	TRUE
14	C	siyou syusei	Timing control correction	FALSE	10	Spec Fault	FALSE
13	C	siyou syusei	Judgment control correction	FALSE	10	Spec Fault	FALSE

要求定義Fault（Define Fault）
コーディングFault（Code Fault）
仕様書Fault（Spec Fault）

(2)最適アルゴリズムの選択

表 10 に示すとおり 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 が最も高いことが分かった．他のアルゴリズムには，J48-C 0.25-M2 よりも高い分類精度を持つものがあるが，10 個のアルゴリズム中 7 位の分類精度であり，絶対値比較としても Ave. 0.963 は 95%以上の信頼区間を得ているため J48-C 0.25-M2 のアルゴリズムは妥当性であるといえる.

表 10 最適アルゴリズムの選択結果

Category	Name	Precision	Rating	Recall	Rating	F-Measure	Rating	ROC Area	Rating	Rating Avg.
trees	J48	0.970	8	0.952	8	0.957	8	0.972	4	7.0
	J48graft	0.950	2	0.937	5	0.941	5	0.874	5	4.3
	RandomForest	0.928	3	0.937	5	0.929	2	0.966	3	3.3
	LADTree	0.900	1	0.921	1	0.905	1	0.369	1	1.0
functions	Logistic	0.937	4	0.937	5	0.937	3	0.983	7	4.8
	MultilayerPerceptron	0.987	10	0.984	10	0.985	10	1.000	10	10.0
bayes	BayesNet	0.977	9	0.968	9	0.971	9	0.997	9	9.0
	BayesSimple	0.950	5	0.937	5	0.941	5	0.986	8	5.8
meta	AttributeSelectedClassifie	0.957	7	0.952	8	0.954	7	0.964	2	6.0
rules	PART	0.957	7	0.952	8	0.954	7	0.983	7	7.3

7. 評価結果

(1)Fault 原因の層別化

図 7 に示す通り，Weka に実装されている木構造の視覚化により，Fault 発生の要因を解析する．再利用ソフトウェアで発生した Fault の中から発生原因を①要求定義 Fault，②コーディング Fault，③仕様書 Fault に層別し再利用ソフトのプロセスの問題点を明らかにした.

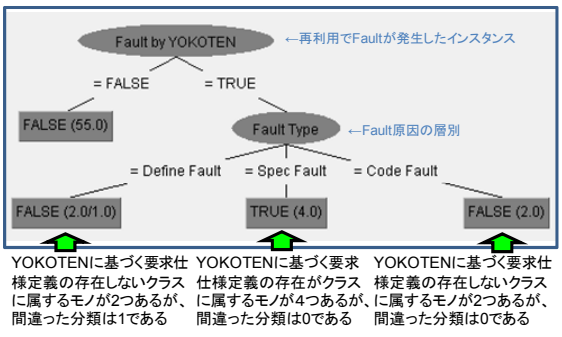


図 7 木構造のビジュアル結果

(2)Fault 発生箇所の読み取り

図 8 に示す通り，再利用先のソフトウェアに発生した Fault の原因の中に，ソフトウェア要求仕様書に基づいたプロセスが存在している場合の分類精度を表している.

①要求定義 Fault 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した.

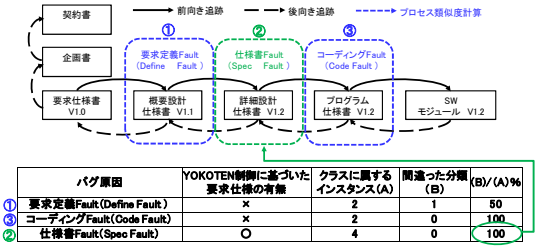
②コーディング Fault は，再利用元に要求仕様の定義が無くても，再利用されたプログラムの Fault を引き継いだ.

③仕様書 Fault 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のプロセスと同じであれば，Fault が発生し得ると考えられる．この結果から，要求仕様の定義に基づき実装しても，ソフトウェアの複雑性に起因する Fault が発生している可能性が明らかとなった.

また，図 8 に示した通り，②仕様書 Fault は概要設計から詳細設計にプロセスが移行するときに，要求された振る舞いを実現できていないために Fault が発生した可能性があることを明らかにした.

これによって，6. 2 (5)の再利用ソフトウェアの Fault の引き継ぎ確認結果から，Fault が発生した再利用元と再利用先との類似度が高ければ Fault 発生の可能性が高まり，また，プロセスに課題があるときは，再利用元のプロセス Fault が再利用先に引き継がれることが分かった.



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Fault(Define Fault)	×	2	1	50
③ コーディングFault(Code Fault)	×	2	0	100
② 仕様書Fault(Spec Fault)	○	4	0	100

図 8 評価結果概要図

8. 今後の課題

本稿の予測手段を適用し，工数低減量を評価する．また開発プロセスはプロダクトの成熟とともに改良されていくため，類似度比較するデータ対象が常に同じであるとは限らない．比較できない追加プロセスは欠損データとなってしまうため，類似度への影響を確認する必要がある.

9. 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プロセスとソフトウェアコードの類似度を比較し，Fault 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．提案方法の有効性を示すために，実データを基にしたメトリクスを用いて類似度によるソフトウェア Fault の発生を特定できることを確認した.

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/.