# 不具合修正時間の要因分析を目的とした例外ルールマイニングの試行

森崎 修司* 森 俊樹** 羽原　寿和** 夏目 珠規子** 山田 淳** 松本 健一*

## An Experimental Evaluation of Unexpected Rule Mining toward Causal Analysis on Defect Correction Time

Shuji Morisaki* Toshiki Mori**, Toshikazu Habara**, Mikiko Natsume**, Atsushi Yamada**, Ken-ichi Matsumoto*

不具合管理表に含まれる個々の不具合の修正時間と属性情報の傾向を抽出し，不具合修正時間に与えている影響の分析を支援する手法を提案する．提案手法では，単一属性ごとの修正時間の可視化だけでは発見の難しい，2 つ以上の属性情報の組合せによる要因分析を支援する．具体的には，修正時間の一般的な傾向を一般ルールとして抽出し，一般ルールと類似しているが修正時間が長くなっているものを例外ルールとして分析者に提示する．本稿では，商用ソフトウェアの開発において蓄積された不具合管理表を対象に提案手法を適用した結果を示し，その有用性を考察する．

This paper proposes an approach for supporting causal analysis of defect correction effort from defect attributes in a bug tracking system. The proposed approach supports analyst by showing relationships between correction efforts and combined attributes as rules. The approach first mines major rules that apply many defects in the bug tracking system. Then the approach mines unexpected rules that are similar to major rules and that have larger correction effort than major rules. This paper shows results of a case study of a bug tracking system in a commercial software development.

Keywords: バグ管理票，傾向抽出，修正工数

## 1. はじめに

ソフトウェア品質向上を目的として，プロセス改善をはじめとする様々な取組みが実施されている．個々の開発メンバの知識をもとに振返り，ブレインストーミング，要因分析等をはじめとして，改善のための方法が提案され，実践されている[1][2]．このような振返りの際に個々の開発メンバの知識だけでなく，定量データをもとに議論ができれば，より根本的な改善や改善の手がかりを効率的に発見できる可能性がある．

不具合の修正時間の要因分析を目的として，不具合管理表に蓄積された修正時間と属性情報を用いることは改善にむけた主要な方法の 1 つであるが，不具合の修正時間と単一の属性情報の関係の分析にとどまることが多い．たとえば，属性 “不具合の混入工程”毎に修正時間をグラフ化すれば，どの工程で混入された不具合の修正時間が長いかを把握することができ，どの工程における品質向上施策を強化すべきかがわかる．しかしながら，複数の属性情報が組合わさった場合にのみ起こる課題や修正時間の長期化は，この方法では把握が難しい．たとえば，システムテストにおいて，特定のモジュールの修正時間が他のモジュールの修正時間よりも突出して長い場合のように複数の属性（モジュール，テスト工程）が組合わさる場合である．

本研究では，修正時間や修正工数に影響を与えている複数の属性情報の組合せをルールの形式で自動的に抽出し，複数の属性情報が組合わさった場合の修正時間の要因分析を支援する．さらに，あてはまる不具合の数が多い傾向を一般ルールとして抽出し，一般ルールに属性が追加されることにより，修正時間が突出して長くなる傾向を例外ルールとして抽出する．一般ルールと例外ルールを対にして分析者に提示することにより，現状把握や課題発見の助けとする．

## 2. 例外ルールによる知見抽出

*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)
** 株式会社東芝　ソフトウェア技術センター
(Corporate Software Engineering Center, TOSHIBA CORPORATION)

選択肢と修正工数（ルール抽出に利用）　　自由記述（不具合の精査に利用）

| ID | 報告日時 | 修正日時 | … | 重要度 | 報告者 | 種別 | … | 修正工数 | … | 症状 | 再現方法 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 11/01/20 | 11/01/29 | … | 重大 | 担当B | 異常系 | … | 3人日 | … | 同時接続・・ | 2つのクライ・・ |
| … | … | … | … | … | … | … | … | … | … | … | … |

図 1　不具合管理票の項目例

2.1 概要

提案手法は，不具合管理表に登録された複数の不具合の間にある不具合の修正時間の傾向を把握することを目的としている．傾向を把握することにより，開発プロセス，開発体制等の課題の抽出につながる．対象となる不具合管理表には，各不具合の修正時間や発見者や深刻度等の不具合の管理情報が含まれている．

不具合管理情報から多くの不具合に共通する不具合の修正時間と不具合の条件を X → y の形式で抽出するとともに，その傾向と類似していながら稀に存在する例外的な条件を X’ → y’の形式で抽出する．特に X と X’が似通っていながら，y と y’が大きく異なるものを比較することにより，問題の発見や現状の把握に役立つと期待される．一般的な条件と例外的な条件を分析者が目視することにより，課題を推測する．条件だけでは推測できない場合には，条件にあてはまる個々の不具合を目視し，開発プロセスや体制に関わる課題を抽出する．

2.2 対象とする不具合管理表

提案手法では図 1 のような不具合管理表を対象とする．1 行が 1 件の不具合に対応し，各列が不具合の情報である．提案手法では，不具合管理表に修正時間をはじめとして修正工数等の修正に必要な労力が記録されていること，選択式の情報，自由記述式の情報の 2 種類の情報に分類できること，を前提としている．

選択式の項目と修正時間は条件を抽出するために用いる．条件はルールの形式で抽出する．不具合管理表が満たすルール全てを抽出すると膨大になるので，出現頻度などの外的基準を与え，有益

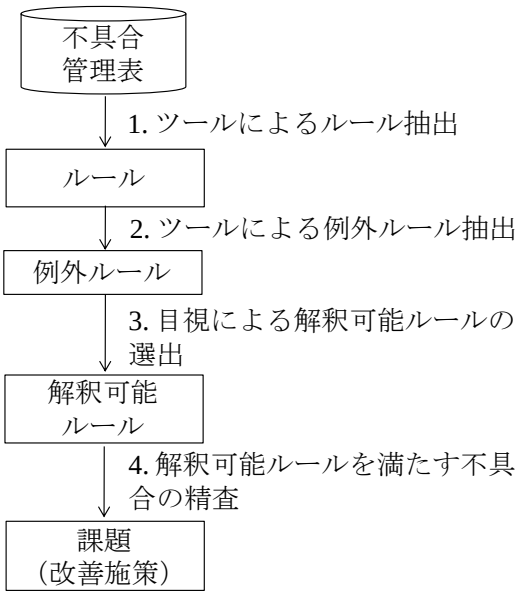


図 2 実施手順

である可能性の高いルールをなるべく残しながら，抽出ルール件数を削減する．図 1 では，選択肢“重要度”, “報告者”, “種別”を例として挙げている．たとえば，重要度は，“重大”，“中”，“軽微”のいずれかの値となる．

自由記述の項目は，ルールを抽出した後，分析者による目視や検討の材料として利用する．たとえば，ルールには “（報告者 =A）→ 修正工数の平均 4.5 人時”というようなものが含まれる場合があるが，報告者 A がどのような不具合を修正しているかは，このルールだけからではわからない．このような場合に，報告者 A が報告している不具合の自由記述部分を読解することにより，その背景や課題が推測できるか試みる．

2.3 手順

提案手法の実施手順を図 2 に示す．

1. ツールによるルール抽出

不具合管理表から文献[3]等の方法を用いて，不

具合管理表に含まれる全ての条件と条件を満たす不具合の修正工数の平均を X→ y の形式の一般ルールとして求める．一般ルールの詳細は後述するが，不具合管理表の項目名（たとえば“重要度”）と項目値（たとえば”重大” ）の組合せを 1 つ以上連結したものである．

2. ツールによる例外ルール抽出

一般ルールと類似の形を持つ例外ルール(X’→ y’)を抽出する．

3. 目視による解釈可能ルールの選出

抽出した一般ルールと例外ルールから，目視により解釈可能なルールを選出する．ここでの解釈可能なルールとは，開発における現状を表わしているものとし，対象ソフトウェアやプロジェクトに十分な知識のある分析者が判断するものとする．

4. 解釈可能ルールを満たす不具合の精査

必要であれば，解釈可能なルールにあてはまる不具合の自由記述を参照し，解釈可能なルールの背景にある課題や改善案を検討する．

2.4 ルール

2.4.1 一般ルール

ルールは X → y の形式で表わし，前提部 X と結論部 y から構成される．X は対象データの項目とその値を「∧」（かつ）で 0 回以上連結したものである．y は対象データのうち，X にあてはまる対象データの間の平均である．X は名義尺度，または，順序尺度の値の組合せである．y は比例尺度の項目である. たとえば, 結合試験で発見され, 深刻度が高い不具合の修正時間の平均が 8 時間であれば，(工程＝結合試験) ∧ (深刻度=高) → (修正時間の平均: 8 時間)と表わす．

前提部を満たす対象データの件数の割合（データ中の出現頻度）を Pr(X → y)と表わす. Pr(X → y)が大きいほど，多くの対象データにあてはまるルールである．

2.4.2 例外ルール

例外ルールは一般ルールと類似のルールでありながら，結論部 y が異なるものであり，一般ルールを X → y としたときに，X ∧ Z → y’となるルールである．y’は y と同じデータ項目であり，値が異なるものを指す．たとえば，一般ルールが「結合試験で発見され，深刻度が高い不具合が平均 8 時間で修正される」であるとき「結合試験で発見され，深刻度が高く，再現性が低い不具合は平均 24 時間で修正される」は例外ルールであり，(工程=結合試験) ∧ (深刻度=高) ∧ (再現性=低) → (修正時間の平均: 24 時間)と表記する．ここで，X は「(工程=結合試験) ∧ (深刻度=高)」, Z は「(再現性=低)」，y は「(修正時間の平均: 8 時間)」，y’は「(修正時間の平均: 24 時間)」である．これらのルールは，項目「(再現性=低)」が加わることにより，修正時間が 3 倍延びている可能性があることを示唆している．

3. 試行

3.1 概要

不具合管理表からプロジェクトの現状を表わすルールが得られるか商用ソフトウェア開発で蓄積された不具合管理表を対象に試行を実施した．対象とした不具合管理表は，結合試験から総合試験までに発見された不具合が登録されている．また，いったん不具合として報告されたが，実際には不具合でないと判断されたものを除き，すべての不具合は修正されている．

2 章で述べた方法により，一般ルールと例外ルールを抽出し，分析者が目視により，現状をあらわしているかどうかを判断した．分析者は対象プロジェクトに関わるメンバとし，ソフトウェアの構成, テストの体制, 不具合管理表の運用ルール, 情報等を把握している．

3.2 対象データ

表 1 に示す項目を持つ不具合管理表から一般ルール，例外ルールを抽出した．修正時間，自由記述以外の項目は選択式の名義尺度，順序尺度であり，ルール抽出に使用した．抽出されたルールが現状を表しているかどうかを判断するために自由記述（不具合の説明，再現方法，修正方法）の情

表 1 対象とした不具合管理表の項目

| 項目名 | 種別 | 値 |
|---|---|---|
| 状態 | 選択肢 | 登録済，振分け待ち，再調査依頼，シミュレーション調査，処置待ち，設計確認待ち，動作確認待ち，処置完了，不具合ではない，保留，仕様調査依頼，調査用テスト実施依頼，動作確認実施依頼 |
| 発見月 | 選択肢 | 1～12 の月 |
| 発見者 | 選択肢 | 担当者 1, 担当者 2, …, 担当者 28 |
| 発生時期 | 選択肢 | 結合試験 1，総合試験 1，システム試験 1，客先試験 1，結合試験 2，総合試験 2，システム試験 2，客先試験 2，結合試験 3，総合試験 3，システム試験 3，客先試験 3，結合試験 4，総合試験 4，システム試験 4，客先試験 4 |
| 優先度 | 選択肢 | 高，中，低 |
| 発生環境 | 選択肢 | シミュレータ, 実機(旧基板), 実機(新基板) |
| 振分者 | 選択肢 | 担当者 1, 担当者 2, …, 担当者 28 |
| 深刻度 | 選択肢 | A, B, C, バグではない，他の報告と重複，仕様変更 |
| 振分先 | 選択肢 | 担当者 1, 担当者 2, …, 担当者 28 |
| 原因モジュール | 選択肢 | Module1, Module2, …, Module49 |
| 処置者 | 選択肢 | 担当者 1, 担当者 2, …, 担当者 28 |
| 修正モジュール | 選択肢 | Module1, Module2, …, Module49 |
| 混入工程 | 選択肢 | 要求分析，基本設計，詳細設計，コーディング，デバッグ |
| 原因 | 選択肢 | 検討不足，ドキュメント上での記載漏れ／コーディング漏れ，ドキュメント上での誤記載／コーディング誤り，誤解／思い込み |
| トリガー | 選択肢 | 起動時，初期化処理時，再起動時，終了時，高負荷時，並列処理時 |
| 設計確認者 | 選択肢 | 担当者 1, 担当者 2, …, 担当者 28 |
| 動作確認者 | 選択肢 | 担当者 1, 担当者 2, …, 担当者 28 |
| 動作確認版 | 選択肢 | Rev.01, Rev.02, …, Rev.85 |
| 再現方法 | 自由記述 | 不具合の再現方法を自然言語で記述 |
| 不具合の説明 | 自由記述 | 不具合の説明を自然言語で記述 |
| 修正方法 | 自由記述 | 修正方法，対処方法を自然言語で記述 |

報を利用した．

3.3 結果

手順 1, 2 で抽出された一般ルールと例外ルールの件数は 32 件である．そのうち，手順 3 で解釈が可能と判断されたルール（1-1, 1-2, 2-1, 2-2, 3-1, 3-2），及び，解釈できなかったルール(4-1, 4-2)を表 2 に示す．1 行が 1 件のルールに対応する．表 2 中の “種別” は一般，例外ルールの種別を表し，“出現頻度”は，ルールに該当する不具合が全体の何パーセントかを表す．“平均修正日数”はルールに該当する不具合の修正日数の平均値である．また，表 2 は，一般ルールの修正時間の平均値と例外ルールの修正時間の平均値の差が大きいものから順番に並べている．

表 2 のルール 1-1, 1-2 は，修正担当者とモジュール，修正日数の関係を表している．ルール 1-1 は担当者 5 が修正するモジュール 6 の不具合の平均修正日数が 10.0 日であり，それが全不具合の

11.2%に該当することを示している．ルール 1-2 はルール 1-1 に「(優先度=中)」が加わることにより，平均修正日数が 7.3 日増加し，この不具合が全体の不具合の 2.2%であることを示している．分析者の解釈は次のとおりである．対象プロジェクトでは，「(優先度=中)」の不具合は他の不具合と比較して，修正の優先順位を低く設定しているため，本ルールは開発の現状を示していると考えられる．

ルール 2-1, 2-2 は不具合修正の優先順位と混入工程，修正日数の関係を示している．ルール 2-1 は，詳細設計で混入した優先度が高い不具合の修正日数の平均が 9.2 日であることを示しており，全不具合の 14.6%に該当することを示している．ルール 2-2 はルール 2-1 に「(発生環境=実機(旧基板))」が加わることにより，修正日数が 7 日増加し，このような不具合が全不具合の 2.8%であることを示している．分析者の解釈は次のとおりである．試験環境はシミュレータを併用しつつ開発の進行に伴い，実機(旧基板)，実機(新基板)の順に移行した．実機(旧基板)でのテストの際に，多くの不具合が発見され，修正が滞っていたことを示していると考えられ，対象プロジェクトの状況を表していると考えられる．ルール 3-1 は発生環境と優先順位，修正日数の関係を表している．

ルール 3-1 は優先度が中であり，発生環境が実機(新基板)である不具合の平均修正日数が 9.5 日であり，全不具合の 13.5%に該当することを示している．ルール 3-2 は，ルール 3-1 に 「(発生時期=システム試験 1)」が加わることにより，修正日数が 5.7 日増加し，このような不具合が全不具合の 3.4%に該当することを示している．分析者の解釈は次のとおりである．本開発では，反復型開発を採用し段階的に機能をリリースした．システム試験 1 は最初のリリースの直前時期であり，「(優先度=高)」の不具合の対応に追われ，特にこの時期において「(優先度=中)」の不具合の修正が滞っており，このルールが対象プロジェクトの状況を表していると考えられる．

一方，プロジェクトの詳細な情報なしにはすぐには解釈ができないルールもあった．ルール 4-1, 4-2 はそのようなルールの一例である．ルール 4-1 は優先度と動作確認版，修正日数の関係を表している．ルール 4-1 は優先度が高く，動作確認版が Rev.26 である不具合の平均修正日数が 9.6 日であり，全不具合の 4.5%に該当することを示している．ルール 4-2 はルール 4-1 に「(発見者=担当者 26)」が加わることにより，修正日数が 4.4 日増加し，このような不具合が全不具合の 2.3%に該当することを示している．これらのルールは分析者によ

表 2 抽出された例外ルールの一部

| | 種別 | ルール | 出現頻度 | 平均修正日数 |
|---|---|---|---|---|
| 1-1 | 一般 | (振分先＝担当者 5) ∧ (修正モジュール＝6) | 11.2% | 10.0 |
| 1-2 | 例外 | (振分先＝担当者 5) ∧ (修正モジュール＝6) ∧ (優先度＝中) | 2.2% | 17.3 |
| 2-1 | 一般 | (優先度＝高) ∧ (混入工程＝詳細設計) | 14.6% | 9.2 |
| 2-2 | 例外 | (優先度＝高) ∧ (混入工程＝詳細設計) ∧ (発生環境＝実機(旧基板)) | 2.8% | 16.2 |
| 3-1 | 一般 | (発生環境＝実機(新基板)) ∧ (優先度＝中) | 13.5% | 9.5 |
| 3-2 | 例外 | (発生環境＝実機(新基板)) ∧ (優先度＝中) ∧ (発生時期＝システム試験 1) | 3.4% | 15.2 |
| 4-1 | 一般 | (優先度＝高) ∧ (動作確認版＝Rev.26) | 4.5% | 9.6 |
| 4-2 | 例外 | (優先度＝高) ∧ (動作確認版＝Rev.26) ∧ (発見者＝担当者 26) | 2.3% | 14.0 |

って解釈ができなかった.

## 4. 考察

対象プロジェクトの不具合管理表から，一般ルール，例外ルールを抽出し，一般ルールと例外ルールの差分をプロジェクトに関わった分析者が目視することにより，プロジェクトの現状を表しているルールを選出することができた. 一般ルール，例外ルールの修正平均日数の差，前提部の項目の差から，不具合修正作業の安定化に影響している変動原因と対策の候補を見つけるなど，開発の現状や課題を抽出できる可能性がある.

今回の試行では，目視による解釈可能ルールの抽出は，対象プロジェクトの状況を詳しく知っているメンバが分析者となって実施した．しかしながら，解釈可能なルールの候補を選出することはプロジェクトの詳細状況を入手できない独立した第三者評価を行う分析者でも実施できる可能性がある．今回の試行では，ルール 2-1, 2-2, 3-1, 3-2 は一般的な開発の知識を持つ分析者であれば，解釈可能なルールの候補として選択できる可能性が高い．ルールの目視を 2 段階に分け，一次選出を一般的なソフトウェア開発の知識を持って第三者評価を行う品質管理担当やプロジェクトを横断して作業を測定分析しているエンジニアリングプロセス改善担当が分析補助者として担当し，二次選出をプロジェクトの状況をよく知る分析者が担当することにより，ルール選出の労力を分担すると共に作業改善のための原因と対策の候補を協議することができる.

一般ルール，例外ルールの抽出条件の設定方法はまだ明らかでなく，今回の試行では，分布を表すヒストグラムや散布図、ルールに含まれる項目など見ながら検討会を行って，試行錯誤を重ねながら設定した．抽出ルールのうちの解釈可能ルールの件数の割合が大きくなるようなルール抽出の外的基準の決定方法は今後の課題である.

## 5. まとめ

不具合管理表に含まれる個々の不具合の登録情報と修正日数の関係を一般ルール，例外ルールの 2 種類の形式で抽出し，分析者の目視によりプロジェクトの現状を表すルールが得られるか実証的に評価した．商用プロジェクトで蓄積された不具合管理表から一般ルール, 例外ルール 32 件を抽出し，プロジェクトに関わっていた分析者により解釈可能なルールを目視で選出したところ，3 対の一般ルール，例外ルールの組合せが選択できた．また，一般ルールと例外ルールの差となる項目の追加が，修正日数を増加させる原因となっているものがあった.

謝辞

本研究は株式会社東芝 ソフトウェア技術センターとの共同研究の一環として実施されたものである．また，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補助費（若手 B : 課題番号 20700028）による助成を受けた.

## 参考文献

[1] D. Card, “Learning from our mistakes with defect causal analysis,” IEEE Software, Vol.15, No.1, pp.56-63 (1998)

[2] R. Chillarege, I. Bhandari, J. Chaar, M. Halliday, D. Moebus, B. Ray and M. Wong, “Orthogonal defect classification-a concept for in-process,” IEEE Trans. on Software Engineering, Vol.18, No.11, pp.943-956 (1992)

[3] 森崎 修司, 門田 暁人, 松本 健一, “ソフトウェアエンジニアリングリポジトリを対象とした例外ルール抽出”, 奈良先端科学技術大学院大学テクニカルレポート No. 20110001 (2011)