レビュー品質の可視化にむけた ODC 分析の応用
－ODC トリガー属性による客観的なレビュー指摘の分類－

Application of ODC Analysis for Visualizing Review Quality
- Objective Classification of Review Comments Based on ODC Trigger Attributes -

日本科学技術連盟 SQiP ODC 分析研究会 チーム 3
Union of Japanese Scientists and Engineers SQiP ODC Analysis Study Group Team 3

○武田 匡広[1)]　小笠原 栄二[2)]　小泉 真一[3)]
○Masahiro Takeda[1)]　Eiji Ogasawara[2)]　Shinichi Koizumi[3)]
玉田 恵子[4)]　田村 真伸[5)]　宮川 真理子[6)]
Keiko Tamada[4)]　Masanobu Tamura[5)]　Mariko Miyakawa[6)]

Abstract

In a V-model-based software development process, conducting high-quality reviews is essential to prevent defects from leaking from upstream to downstream processes. However, while reviews have been evaluated in terms of efficiency, such as the number of issues identified per hour or per page, a method for assessing the quality of the issues themselves has not been established.

Therefore, our research team considered using the "trigger" attribute of Orthogonal Defect Classification (ODC) as a means to evaluate the quality of reviews. For this approach to work, it is necessary that the issues can be categorized using trigger attributes.

As a preliminary step towards evaluating review quality with trigger attributes, our research team devised a method for accurately selecting triggers. Through experimentation, it was confirmed that this proposed method allows for objective selection, free from subjectivity.

1. はじめに

ソフトウェア開発における生産性向上において，上流工程から下流工程への欠陥流出を防止することの重要性はよく知られている[1][2][3].

V字モデルを前提としたソフトウェア開発プロセスの場合，基本設計から詳細設計工程

1） オリンパスメディカルシステムズ株式会社，Olympus Medical Systems Corp.
2） 東芝電波テクノロジー株式会社，TOSHIBA ELECTRONIC TECHNOLOGIES CORPORATION
3） アルプス システム インテグレーション（株），Alps System Integration Co., Ltd.
4） キヤノン IT ソリューションズ株式会社，Canon IT Solutions Inc.
5） 株式会社コムニック，COMNIC CORPORATION
6） 株式会社 構造計画研究所，KOZO KEIKAKU ENGINEERING Inc.

が上流工程に相当し，これら工程からの欠陥流出を防止するにはレビューの実施が有効である[5]．すなわち生産性向上のためにはレビュー品質の向上が重要であり，そのためにはレビュー品質の可視化が必要となる[4][5]．

しかし，レビューについては「時間あたりの指摘数」や「ページあたりの指摘数」など効率面で定量的に品質を分析することはできても指摘の内容そのもの，すなわち「指摘の質」に対する定量的な分析を行うことが難しい．

そこで筆者ら研究チームは「指摘の質」を評価するためソフトウェア欠陥分析手法の 1 つである ODC 分析[6][7]を応用できると考えた．特に ODC 分析で用いる分類属性の中でも欠陥が検出された経緯に着目する「トリガー属性」を用い，「レビューで得た指摘に対するトリガーの傾向」と開発工程終盤で実施されるシステムテストで検出された「レビュー漏れに起因する欠陥の量」との相関を見ることでレビューの質が評価できるとの仮説を立てた．すなわちレビューにおけるトリガーの傾向とレビュー漏れにより後工程へ流出した欠陥量に相関が得られれば，レビューにおけるトリガーの傾向により後工程への欠陥流出量が予測でき，その予測された流出量の多寡によりレビューの質が評価できるというものである．

一方で ODC 分析をレビューに適用した事例はまだなく，レビューで得た指摘を適切にトリガー属性で分類する手法も確立してはいない．実際，筆者ら研究チームにて既存のレビュー記録についてトリガー属性による分類を行ったところ，研究員の間でも選択するトリガーにバラツキがみられた．これは人により指摘内容の着目点，すなわち「気になった点」の選び方が主観によるためと考える．そこで筆者ら研究チームは次の二点を軸に人の主観に依存しにくいトリガー選択手法を考案した．

- レビュー記録の何処を読み取り，何故そのトリガーを選択したのか，その思考の過程を記録に残す．
- 抽出された思考の過程を可視化し再現可能な状態とすることで，トリガー選択の曖昧さを除去する．

本研究では，レビューで得た指摘に対してトリガーを選択する際に，考案した手法を用いることにより，人の主観が排除され選択のバラツキが減少することを確認する．すなわち本研究における Research Question は以下である．

RQ：提案手法を適用することでレビューに対するトリガー選択の属人性を排除することができるか．

以降，第 2 章では筆者ら研究チームが考案したトリガー選択手法を示す．続く 3 章，4 章にて実験方法とその結果を示し，最後に 5 章と 6 章にて考察と今後の展望を述べる．

2. 提案する手法

2.1 トリガーについて

提案する手法を述べるにあたりODC分析におけるトリガー属性について簡単に述べる．トリガー属性とはテストで検出された欠陥を表す性質の一種であり，その欠陥が「どの様な手順で検出されたか」という点に着目したものである．例えば結合テストおよびシステムテスト工程において，トリガーはより簡単な手順からより複雑な手順へと 11 種類が定義されており（表 1)，原則として後者に分類される欠陥が多いほどよりテストが成熟されている，すなわちテストが十分にやり尽くされていると考えられている．

表 1 テストにおけるトリガー[8]

トリガー名	意味
基本	1 つの機能をデフォルトの状態で実行した結果，欠陥を検出した場合.
バリエーション	1 つの機能をオプション/パラメータを変更して実行した結果，欠陥を検出した場合.
実行順序	複数の機能を順番に実行した結果，または単一機能を複数回実行した結果，欠陥を検出した場合.
相互作用	2 つ以上の機能を同時に実行した結果，欠陥を検出した場合.
正常系	ある意図（異常系のテスト．「負荷/ストレス」～「SW 構成」に相当）をもってテストを実施したところ，それ以前の基本的な欠陥（結合テストで検出されるべき欠陥）が検出された場合.
負荷/ストレス	リソースに意図的な負荷をかけるテストを実行した結果，欠陥を検出した場合.
復帰/例外	例外処理やリカバリーを試すテストケースを実行した結果，欠陥を検出した場合.
起動/再起動	シャットダウン，スリープ，ネットワーク遮断，電源遮断，再起動（異常状態から通常状態に戻る）の過程をテストした結果，欠陥を検出した場合.
HW 構成	ハードウェア構成の組み合わせに関するテストを実行した結果，欠陥を検出した場合.
SW 構成	ソフトウェア構成の組み合わせに関するテストを実行した結果，欠陥を検出した場合.
修正確認	欠陥が修正されたことを確認している際に，修正の副作用である新規欠陥を検出した場合.

レビューにおいても同様に 8 種類のトリガーが定義されている（表 2)．レビューにおけるトリガーはレビューで得た指摘，すなわちドキュメント内の誤りについて「どのような観点でレビューをすればその誤りに気がつけるか」を示したものとなるため，トリガーの分布傾向はレビュー観点の多様性を示唆する．ただし「どの様な分布であればレビューがやり尽くされたと言えるか.」は今後の研究課題である．

表 2 レビューにおけるトリガー[8]

トリガー名	意味
後方互換性	広範囲な製品知識や経験を活かし，レビュー対象と前バージョンソフトウェア，あるいはプロトタイプとの間に食い違いがないか確かめたときに，欠陥を検出した場合.

水平互換性	広範囲な製品知識や経験を活かし，レビュー対象と他システムや製品，サービス，コンポーネント，モジュールなど相互作用するものとの間に食い違いがないか確かめたときに，欠陥を検出した場合.
稀な状況	広範囲な経験や製品知識を活かし，レビュー対象であるコードや設計からは予測が付きにくい振る舞いを予見し，その結果，欠陥を検出した場合. 大抵の場合，それらは想定外の構成や操作で引き起こされる．なお，エラー回復機能の欠損や誤りは，一般的にはここに分類しない.
デザイン整合性	レビュー対象が，前工程の成果物である仕様書や設計書，あるいは社内の開発標準や国際規格等の内容と合致していることを確かめたときに，欠陥を検出した場合.
ドキュメント一貫性	ドキュメントレビューに於いて，ドキュメント内の記述に一貫性があるか（内容が異なっていないか，矛盾がないか）を確かめたときに，欠陥を検出した場合.
開発言語依存	実装されたコンポーネントや関数について，開発言語仕様を確認（言語標準に沿って記述されているか，言語固有の効率的な記述ができているかなど）した結果，欠陥を検出した場合.
詳細理解	コンポーネントの構造や動きを詳細に理解しようとしたときに，欠陥を検出した場合．特に次の観点で見た場合. a.機能の実装に要求されるロジックやフローを検証した場合. b.並行処理や複数同時稼働について検証した場合.
修正確認	仕様変更の結果，その妥当性を再レビューしている際に，その変更による副作用である新規欠陥を検出した場合.

2.2 トリガー選択のフローチャート

筆者ら研究チームはチーム内で収集したレビューで得た指摘約 60 件について実際にトリガーを選択し，その選択の際の思考の過程を言葉で表した．その言語化された思考の過程より最終的に何がトリガー選択の決め手となったのかを抜き出し，トリガーを決めるために必要な要素を 8 つの質問という形にまとめた（表 3）．ただし質問形式では回答する順序によりトリガーが変わることがあるため，より具体的な質問からより抽象的な質問に解答できるよう質問する順序を決めて，最終的にそれをフローチャートとして完成させた（図 1）．なおレビューのトリガーでは誤字脱字等の軽微な誤りは欠陥として識別しないがフローチャート上ではオプションとして取り入れた.

表 3 レビュー向けトリガーを決めるための質問

質問	対応するトリガー
仕様変更や欠陥修正に伴う変更に起因する指摘か？	修正確認
前バージョンやプロトタイプなどとの不整合に関する指摘か？	後方互換性
相互に作用する製品などとの不整合に関する指摘か？	水平互換性
その他，たまたま知っていた知識や情報などに基づく指摘か？	稀な状況
対象ドキュメントへのインプットや，規格，標準などとの不整合に関する指摘か？	デザイン整合性
用語や定義の不一致などドキュメントの内の矛盾や，曖昧な情報に関する指摘か？	ドキュメント一貫性
開発言語や OS 等の仕様に依存する指摘か？	開発言語依存
処理の考え方や作りに関する指摘か？	詳細理解

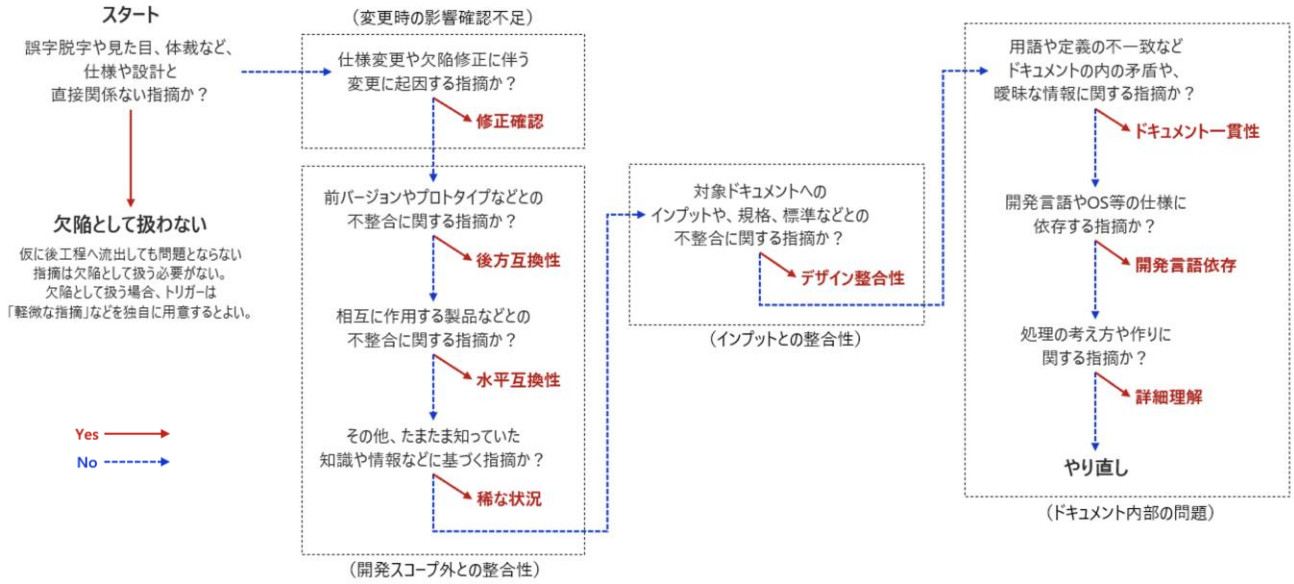

図 1 レビュートリガー選択フローチャート

3. 実験方法

フローチャートを用いることによりトリガー選択の属人性が排除されることを検証するため，筆者ら研究チームとは別の 5 名の被験者に対して検証を行った．その手順を以下に示す．なお 5 名はいずれも ODC 分析研究会のメンバーであり ODC 分析についての知識と経験を有してはいるが，レビューのトリガーについての利用経験はない．なお実験に用いた指摘について内容は非公開とするが，代わりにその概要を表 4 に示す．

(1) 1 回目：実際のレビュー指摘 30 件について，5 名がそれぞれトリガーを選択する．その際，フローチャートは適用しない．

(2) 2 回目：別の指摘 30 件について，同 5 名がそれぞれトリガーを選択する．その際，フローチャートを適用する．なお，2 回目は 1 回目の試行による慣れの影響を排除するため二週間のインターバルを置いた後に実施する．

各回について回答のバラ付き方を調べ，フローチャートを適用した場合は回答のバラツキが収束することを確認する．なおバラツキは 5 名の回答の一致度，すなわち何人が同一トリガーを選択したかリッカート尺度を用いて 7 段階で表現する．実際に用いた尺度を表 5 に示す．

表 4 レビュー指摘記録の概要

内容	1 回目	2 回目
対象工程	基本設計	基本設計
指摘件数	30 件	30 件
指摘一件あたりの平均情報量	約 94 文字	約 118 文字

表 5 トリガー選択一致度

一致度	表記	意味
7(最高)	5	全員が同じトリガーを選択した
6(高い)	4：1	選択が二群に分かれたが，そのうち 4 人は同じトリガーを選択した
5(やや高い)	3：2	選択が二群に分かれたが，過半数が同じトリガーを選択した
4(普通)	3：1：1	選択が三群に分かれたが，過半数が同じトリガーを選択した
3(やや低い)	2：2：1	選択が三群に分かれており，どれも過半数に達していない
2(低い)	2：1：1：1	選択が四群に分かれており，どれも過半数に達していない
1(最低)	1：1：1：1：1	全員が異なるトリガーを選択した

4. 結果

レビューで得た指摘に対してトリガーを選択した際の，フローチャートの適用有無による結果の違いを図 2 に示す．

1 回目（フローチャートなし）では 30 件中 7 件が一致度 4（普通）となり，ここをピークに一致度が高い方と低い方に分散した．2 回目（フローチャートあり）では，30 件中 8 件が一致度 7（最高）となり，ここをピークに右肩下がりの傾向となった．一方で一致度 1（最低）となる指摘が 2 回目でも 1 件確認された．

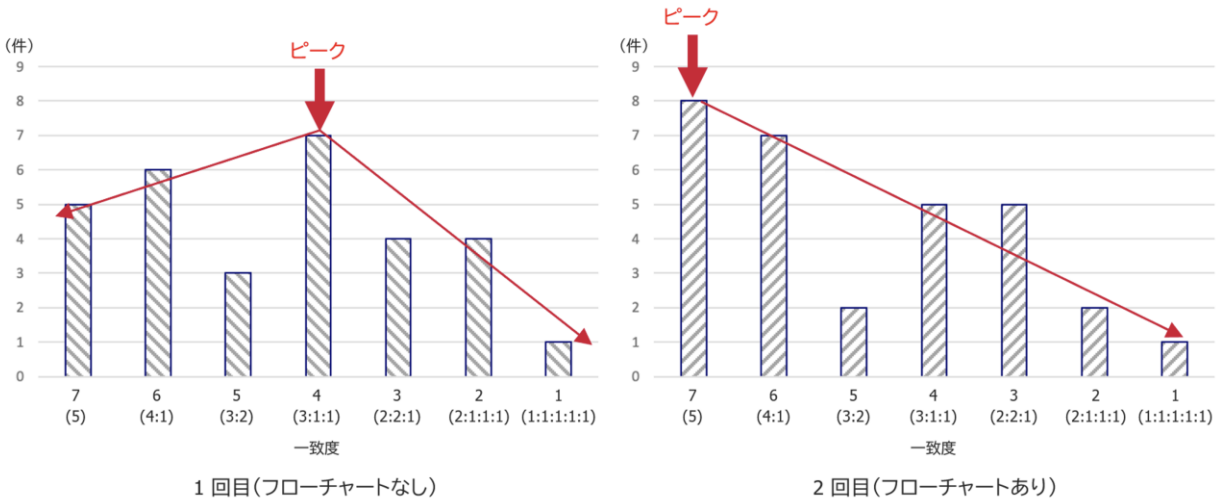

図 2 フローチャートの適用有無によるトリガー選択の結果

5. 考察

本研究の RQ である「提案手法を適用することでレビューに対するトリガー選択の属人性を排除することができるか」について，実験結果を踏まえ考察を述べる．

今回，筆者ら研究チームはレビューにおけるトリガー選択について，その属人性を排除すべくトリガー選択用のフローチャートを考案した．実験によりトリガーの選択にフロー

チャートを適用した場合，そうでない場合と比較して一致度のピークが 4（普通）から 7（最高）へ移動し，全体傾向も最高から最低へ右肩下がりとなることが確認できた．

その一方でフローチャートを適用したにもかかわらず，一致度が 1（最低）となるケースも 1 件検出された．この 1 件ついて内容を確認したところ，設計の誤りに関わる指摘ではなく情報量の不足について説明を求めるものであり，そのため適切なトリガーが選択できなかったと考えられる．これについては選択肢に「わからない」を追加する等の対策が必要である．

また，1 回目と 2 回目では指摘 1 件あたりの情報量が異なり 2 回目の方が多い（表 4）．この情報量の差が一致度に影響したか確認するため，指摘 1 件あたりの情報量を一致度 7〜5（やや高い以上），4（普通），3〜1（やや低い以下）の 3 層に別け，情報量と一致度の関係を確認した（表 6）．結果，必ずしも情報量の多さが一致度の高さと関係あるとは言えないことを確認した．

表 6 一致度別の情報量

内容	1 回目	2 回目
一致度 7〜5（やや高い以上）の指摘 1 件あたりの平均情報量	約 99 文字	約 113 文字
一致度 4（普通）の指摘 1 件あたりの平均情報量	約 81 文字	約 93 文字
一致度 3〜1（やや低い以下）の指摘 1 件あたりの平均情報量	約 94 文字	約 145 文字

今回の実験では試行の回数が十分でないことは否めないが，上記実験の結果を以て RQ「提案手法を適用することでレビューに対するトリガー選択の属人性を排除することができるか」については，「排除は可能である」と結論する．

6. 今後の展望

レビュー品質の可視化，とりわけ指摘の質の可視化という課題に向けて，まずは指摘を ODC 分析の属性の 1 つである「トリガー」で分類できるかを検証した．その結果，トリガーを選択する思考の過程を表現したフローチャートを用いることにより属人性を排除した分類が可能であることが確認された．これにより，従来の「時間あたりの指摘数」，「ページあたりの指摘数」など効率面に着目したレビュー品質の可視化に加え，「指摘の質」の評価の実現に目処が付いたと考える．今後の展望として，次は指摘の質の評価を実現するため以下の検証を進めていく．

(1) レビューの結果からシステムテストの結果を予測する

「レビューで得た指摘に対するトリガーの傾向」と開発工程終盤で実施されるシステムテストで検出された「レビュー漏れに起因する欠陥の量」との相関を確認する．もし両者の間に何らかの相関が得られれば，レビューが完了した段階でその指摘をトリガーで分類し，その傾向を見ることでシステムテストの結果を予測できると考

える.

(2) システムテストの結果よりレビューの弱点を推定する
逆にシステムテストで検出された欠陥のうち，上流工程でのレビュー不足に起因する欠陥について，どのトリガーで指摘されるべきだったかを考える．これによりレビュー時に不足した観点を知り，レビューのやり方について改善を促すことが可能と考える．

参考文献

[1] 丸山, 志保., & 柳田, 礼子. (2017). レビュー重視と品質・生産性の関係分析. *ソフトウェア品質シンポジウム 2017*.

[2] Kemerer, C., & Paulk, M. (2009). The impact of design and code reviews on software quality: An empirical study based on PSP data. *IEEE Transactions on Software Engineering, 35*(5), 534-550

[3] Fagan, M. E. (1976). *Design and code inspections to reduce errors in program development. IBM Systems Journal, 15*(3), 182-211.

[4] Meyer, B., & Hannan, M. (2007). *The Role of Visualization in Improving Review Quality: A Case Study. IEEE Transactions on Software Engineering, 33*(6), 391-402.

[5] Ciolkowski, M., Laitenberger, O., & Biffl, S. (2003). Software reviews: The state of the practice. *IEEE Software, 20*(3), 46-51.

[6] 杉崎, 眞弘., & 佐々木, 方規. (2020). *ソフトウェア不具合改善手法 ODC 分析*. 日科技連出版社.

[7] Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Moebus, D. S., Ray, B. K., & Wong, M.-Y. (1992). Orthogonal defect classification: A concept for in-process measurement. *IEEE Transactions on Software Engineering, 18*(11), 943-956.

[8] ODC 分析研究会. (2024). *ODC 分析研究会版 ODC 属性第 1.0 版*.