リファクタリングによる自然さの変化に関する調査 ―自然さによるリファクタリング支援を目指して―

有馬 諒[1,a)]　肥後 芳樹[1,b)]　楠本 真二[1,c)]

概要：リファクタリングはソフトウェアの保守性を向上させるための重要な作業であるが，ソースコード中のどの部分にどのようなリファクタリングを適用するかは，開発者の経験や勘によるものが大きく，リファクタリングを行うことを難しくしている．そこで本研究では，近年盛んに研究されているソースコードの自然さをリファクタリング支援に用いることを目指し，実際のオープンソースソフトウェア開発で行われたリファクタリングに対して，その前後での自然さの変化を調査した．その結果，調査対象の 28 個のうち 19 個のリファクタリングでは，自然さが向上していたことを確認した．このことから，自然さはリファクタリングによって変化するソースコードの特性をとらえており，自然さをリファクタリング支援に用いることは有効であると考えられる．

1. はじめに

リファクタリングとはソフトウェアの外部的振る舞いを保ちながら内部構造を改善する作業であり，ソフトウェアの保守性を向上させるために重要な作業である．しかし，ソースコード中のどの部分にどのようなリファクタリングを適用するかは，開発者の経験や勘によるものが大きく，リファクタリングを行うことを難しくしている．

本研究では，ソースコードをその “自然さ” という観点から評価することによってリファクタリングを支援する手法を検討する．自然さとは，自然言語処理の手法である言語モデルを用いて求められる指標であり，単語列がその言語の文としてどれだけ自然かを数値で表したものである．頻繁に出現する単語列ほど高い自然さが割り当てられ，反対にほとんど出現しない単語列の場合低い自然さが割り当てられる．ソースコードの自然さに関する研究は近年盛んに行われており，コード補完 [1] や，バグ検出 [2] などで成果が報告されている．

本研究ではソースコードの自然さを用いたリファクタリング支援研究の第一歩として，自然さをリファクタリング支援に用いることが有効かを検証するための調査を行った．この調査では実際のオープンソースソフトウェア開発で行われたリファクタリングに対して，その前後での自然さの変化を調査した．その結果，対象の 28 個のリファクタリングのうち，19 個のリファクタリングによってソースコードの自然さが向上していたことを確認した．このことから，自然さはリファクタリングによって変化するソースコードの特性をとらえており，自然さをリファクタリング支援に用いることは有効であると考えられる．

[1] 大阪大学 大学院情報科学研究科
[a)] r-arima@ist.osaka-u.ac.jp
[b)] higo@ist.osaka-u.ac.jp
[c)] kusumoto@ist.osaka-u.ac.jp

2. 調査

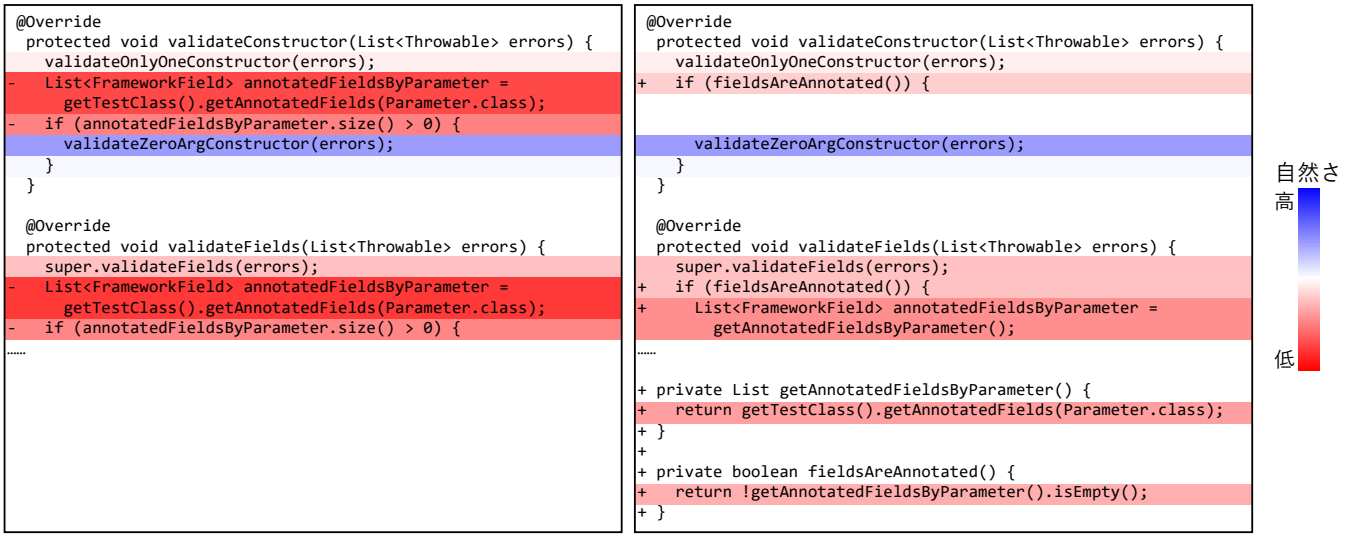
本研究では，自然さをリファクタリング支援に用いることが有効かを検証するため，オープンソースソフトウェア開発において実際に行われたリファクタリングによって自然さがどう変化しているかを調査した．

2.1 調査方法

本研究の調査ではまず，自然さ計測のための言語モデルを構築した．本研究では，Tu らの研究 [3] によって用いられたキャッシュ付き n-gram 言語モデルを用いた．モデル推定のためのソースコードコーパスには，Higo らによって作成されたデータセット [4] を用いた．これは Apache Software Foundation[*1]の Java プロジェクトから作成されたデータセットであり，テストコードや自動生成ファイルが取り除かれた 66,724 Java ファイル，11.5 MLOC からなる．

次に，調査対象となるリファクタリングを収集した．調査対象のオープンソースソフトウェアとして，JUnit4[*2]を用いた．JUnit4 は GitHub によって管理されており，GitHub

[*1] http://www.apache.org/
[*2] https://github.com/junit-team/junit4



(a) リファクタリング前　　(b) リファクタリング後

図 1　コミット#0215c66 における自然さの変化．+で始まる行はリファクタリングによって追加された行を，-で始まる行はリファクタリングによって削除された行を表す．

を用いた開発者相互によるソースコードレビューによってソースコードの品質が担保されている．本研究ではまず，JUnit4 リポジトリからコミットメッセージにリファクタリングに関連があると思われるキーワードを含むコミットを抽出した．本実験ではキーワードとして *refactor*，*clean* を用いた．次に抽出されたコミットを目視で確認し，変更内容がリファクタリングのみであり，処理の内容を変えないコミットのみを抽出した．こうして抽出した 28 コミットを調査対象として用いた．

最後に，抽出した各リファクタリングに対して自然さの変化を調査した．各リファクタリングにおいて，リファクタリング適用前と適用後の自然さを構築した言語モデルによって計測することで，リファクタリングによって自然さが向上したか，低下したかを調査した．

2.2　調査結果

調査の結果，28 個のリファクタリングのうち 19 個では自然さが向上していた．この結果から，自然さはリファクタリングによって変化するソースコードの特性をとらえており，自然さをリファクタリング支援に用いることは有効であると考えられる．

自然さが向上した例として，コミット#0215c66 で行われたリファクタリングを図 1 に示す．図において赤い行は自然さが低い行を，青い行は自然さが高い行を表す．このコミットでは複数箇所で共通する処理をメソッドとして切り出すリファクタリングが行われている．その結果，自然さの低かった行がメソッド呼び出しに変更されることで自然さが向上し，ソースコード全体の自然さが向上したと考えられる．

3.　おわりに

本研究では，ソースコードの自然さを用いたリファクタリング支援研究の第一歩として，実際に行われたリファクタリングにおいて自然さがどう変化しているかを調査した．その結果，オープンソースプロジェクトに対して行われた 28 個のリファクタリングのうち，19 個のリファクタリングにおいて自然さが向上しており，自然さをリファクタリング支援に用いることは有効であると考えられる．今後はこの結果を踏まえて，自然さを用いたリファクタリング支援手法を開発する予定である．

謝辞　本研究は，科学研究費補助金基盤研究 (B)(課題番号：17H01725) の助成を得て行われた．

参考文献

[1] Hindle, A., Barr, E. T., Su, Z., Gabel, M. and Devanbu, P.: On the Naturalness of Software, *Proceedings of the 34th International Conference on Software Engineering*, pp. 837–847 (2012).

[2] Ray, B., Hellendoorn, V., Godhane, S., Tu, Z., Bacchelli, A. and Devanbu, P.: On the “Naturalness” of Buggy Code, *Proceedings of the 38th International Conference on Software Engineering*, pp. 428–439 (2016).

[3] Tu, Z., Su, Z. and Devanbu, P.: On the Localness of Software, *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 269–280 (2014).

[4] Higo, Y. and Kusumoto, S.: How Should We Measure Functional Sameness from Program Source Code? An Exploratory Study on Java Methods, *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 294–305 (2014).