5ZF-4

オブジェクト指向プログラミング初学者向け評価手法を用いたソースコード改善学習支援

若林　智徳†　　松浦　佐江子‡

芝浦工業大学大学院工学研究科　電気電子情報工学専攻†‡

1. はじめに

ソフトウェア開発の規模の拡大に伴い，プログラムが含む問題は複雑化している．このため，ソフトウェア品質特性（ISO/IEC9126-1:2001[1]）の一つである保守性の高いソースコードを開発することの重要性が高まっている．リファクタリング[2]は，ソースコードの保守性を改善する一つの有効な手段である．保守性の高いソースコードを定義することは，プログラムを適切に設計する能力とも関係することから，プログラミング学習者に，早期からソースコードの品質の概念やリファクタリングの手法を習得させることが望ましい．リファクタリングの効果は，ソフトウェアメトリクス[3]等を用いることにより，数値的に評価することも可能であるが，各メトリクスの値は，ソースコードの様々な問題と 1 対 1 に対応しているわけではないため，リファクタリングすべき箇所をメトリクスの値のみから直接特定することは難しい．これら小さな問題が未解決のまま放置されると，より規模の大きい問題を解決する際の障害となる．CheckStyle や FindBugs 等の既存の評価ツールでは，ソースコードの標準適合性や，問題発生の可能性が高いコードに対して警告を行えるが，解決の方法や優先順序は具体的に示されないため，学習されないことがある．

本研究のねらいは，オブジェクト指向プログラミング学習者に対し，簡易で小規模なリファクタリングを実践させることで，ソースコードの品質改善を自力で行う能力を養成することである．本稿では，初学者が認識していない問題点に対して解決しやすい順序でリファクタリングするプロセスを，問題点の判定に必要な品質評価観点や要素，改善方法に基づき学習する支援ツールと，その適用効果について考察する．

2. ソースコードの品質評価

2.1 初学者ソースコードの問題分析

我々は，初学者のソースコードに含まれる問題について分析を行った[4]．プログラミング教育においては，機能的な評価だけでなく品質評価も行うことで，変更要求やバグに迅速に対応できるソースコードを記述できる能力を養うことができる．そこで，品質評価に必要な要件について考える．リファクタリング技能を習得させるために必要な要件は，ソースコードを理解させることと，クラスの責務を正しく認識させることである．クラスの責務とは，クラスが果たす役割を指し，その構成要素であるフィールドとメソッドで表される．ソースコードを読みやすくし，クラスの責務を修正する知識はプログラムの設計段階での問題の発生を未然に防ぐことに繋がる．上述の 2 点について，次の 7 つの観点に着目する．

Source Code Refactoring Support Tool for Novice Object Oriented Programmer

†Tomoyoshi Wakabayashi ‡Saeko Matsuura

†‡Department of Electronic Information Systems, Graduate School of Engineering, Shibaura Institute of Technology.

2.2 品質評価観点

・未使用な構成要素

プログラム実行時に使用されないクラスやコンストラクタ，メソッド，フィールド，ローカル変数を指す．これらは，コーディングの過程で不要となった構成要素が削除されずに残されたものである．これらが存在するソースコードは，その要素が不要であるかを判断する余計な手間を作り，記述内容の理解を妨げる要因となる．

・未定義のドキュメンテーションコメント

ドキュメンテーションコメントは，クラスやメソッド，フィールドに対して，それが定義された意図を説明するために記述される．説明する要素の直前に定義しなければならないという制約があるため，説明する対象が明確にされている．従って，これを記述させることによって，クラスの構成要素を定義した意図を考えさせる動機を作り，不要な構成要素の定義を防ぐことができる．

・マジックナンバーの使用

ソースコード上に記述された数値は，定数として宣言することにより，定義した意図を定数名として明示的に表すことができる．特に if 文や for 文といった制御構文の条件文中等で，比較演算子のオペランドに数値が含まれている場合，何らかの閾値である場合が多い．これらを定数化することで，何の閾値であるかを明確にする．

・1 ソースファイルによるクラス管理

オブジェクト指向言語においては，1 ソースファイルには 1 クラスのみ定義されるべきである．複数のクラスが定義されたソースファイルは，あるクラスのみを利用しようとした場合でも，そのソースファイルに含まれる全てのクラスと，それらクラスが依存するクラスを読み込むため，余分なクラスを含むプログラムとなる．

・与えられた役割を果たさない変数

あるメソッドのローカル変数としてのみ利用される変数は，フィールドとして宣言されるべきではなく，また，ループ変数としてループブロック中でのみ使用される変数が，ループブロック外で宣言されるべきではない．これらは，使用される直前で宣言する．

・可視性の緩い構成要素

フィールドやメソッドの可視性の定義が適切な範囲でない場合，フィールドに対して他のクラスが自由にアクセス可能となり，プログラムが誤動作する原因となる．クラスの外部へ見せるべきではない構成要素は，可視性を適切に定義し保護しなければならない．

・不適切なクラスの責務分割

不適切なクラスの責務分割は二つの副観点を持つ．一つは，クラスが本来意図された責務から考えて持つべきではない振る舞いや，同じクラス内で関連性の無い振る舞いを持つ場合であり，その振る舞いを適切なクラス，あるいは新しいクラスに移動する必要がある．もう一つは，クラスが本来持つべき責務が与えられておらず，データ保持のみのクラスである場合であり，他のクラスか

ら必要な振る舞いを移動，または作成する必要がある．

3. 学習支援ツール

3.1 品質評価ツール

品質評価観点に基づいて，ソースコードを評価する品質評価ツールを実装した．品質評価ツールは，まず，ASTParser を利用して，入力された Java ソースコードから構成要素を抽出する．次に，品質評価観点別に評価対象と判定に必要な構成要素を抽出する．これらの要素から評価対象毎に問題の有無を調べる．発見された問題には，解決の優先順位を設定し，複数の問題が絡んだ場合は，この順位を基に解決を行う．最後に，評価対象毎に集計を行い，HTML 形式のレポートを出力する．レポートには，問題の発生箇所，要素名，問題名，問題の概要と修正方法を記述する（図 1 下部）．

3.2 学習環境

リファクタリング技能を習得するために必要な要件として，コードエディタと連動して問題箇所が強調され，ソースコードに合わせて動的に改善方法が提示される仕掛けが求められる．このような要件を満たす環境として，Web 教科書[5]を利用する．Web 教科書は，講義毎にテーマに沿った解説や，コードエディタを利用したコード問題を課すことが可能な e-Learning システムである．コードエディタはコンパイルや実行，ソースファイル管理を行う機能を持ち，アドオンとして前章の品質評価ツールを組み込むことができる．Web 教科書を利用して，各品質評価観点について，何が問題であり，この問題を放置するとどのような不具合が起きるのかを説明し，その修正方法を例題を用いて実習させる．Web 教科書を利用することによって，小さな問題については学習者自身で解決できるようになり，教員はより複雑な問題の修正方法を，時間をかけて教示することができる．図 1 は Web 教科書環境での実装例である．上部右側にコードエディタがあり，下部は評価結果レポートが表示されている．ソースコードの問題箇所にはハイライトが付けられ，クリック動作で対応する評価結果レポートが表示される．

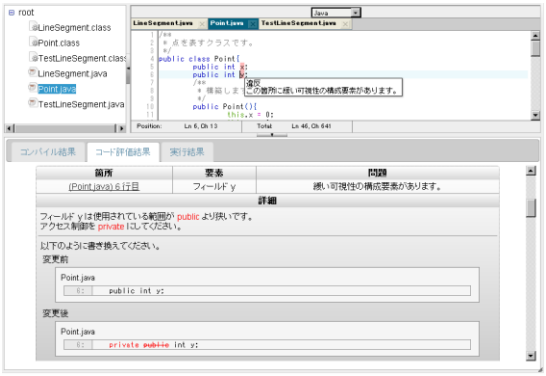

図 1 Web 教科書環境での実装

4. 品質改善学習実験

4.1 実験プロセス

本学学部 3 年生 8 名を対象に 3 日間のソースコード品質改善講習を行った．講習内容は，2 章の品質評価観点に基づいたもので，実習は 3 章の Web 教科書上で動作する品質評価ツールを用いた．講習後，全ての品質評価観点の問題を含むソースコードを配布し，それらの問題を解消するようにリファクタリングを行うという内容の課題を課した．配布したソースコードは，後置記法で記述された式を計算し標準出力に表示する 5 クラス 270 行程度の小規模なプログラムである．課題の解決のヒントとして，問題箇所をハイライトするのみの品質評価ツールを提供し，任意で使えるようにした．

4.2 結果と考察

課題を提出した 6 名について，表 1 に品質評価観点別の正答数をまとめた．評価数は観点毎に評価を行った箇所の数であり，有効正答数は，出題時に想定した修正が行われた箇所の数である．有効正答率を見ると，ソースファイル管理については，全員が正答しているが，それ以外の観点については正答率が 50%を下回っており，問題の見落としや修正方法が分からなかった箇所が多かったと考えられる．特に，与えられた役割を果たさない変数の正答率が悪く，変数のスコープに対する認識が不十分であることが窺える．品質評価ツールを利用した学生の正答率は比較的高く，利用していない学生は，未使用や可視性の緩い構成要素の見落としが多かった．コードを眺めただけでは気付きにくい問題については，ツールで支援することにより，見落としを減らすことが期待できる．修正されたソースコードを見ると，複数の責務を持つクラスを分割させた場合，新しく作成したクラスに移動したフィールドの可視性が再考されていない，未使用なフィールドを削除していない等，問題が複数重なっている箇所での見落としがあった．また，データクラスの修正では，フィールド自体の隠蔽はできているが，新しく定義したメソッドにマジックナンバーが含まれ，新たな問題が作られる現象があった．

表 1 品質評価観点別の正答数

品質評価観点	評価数	有効正答数	有効正答率
未使用な構成要素	30	9	30.0%
未定義のドキュメンテーションコメント	102	26	25.5%
マジックナンバーの使用	18	4	22.2%
1ソースファイルによるクラス管理	6	6	100%
与えられた役割を果たさない変数	42	7	16.7%
可視性の緩い構成要素	42	13	31.0%
不適切なクラスの責務分割(複数の責務)	24	8	33.3%
不適切なクラスの責務分割(データクラス)	18	6	33.3%

5. まとめ

本研究では，品質評価観点に基づく品質評価ツールを実装し，講習会形式で品質改善実験を行ってその効果を検証した．ソースコードの保守性の低さは，小さな問題の見落としの蓄積により表面化するが，ツールを使って支援することにより早期に修正を行わせることができた．

今後の課題として，見落としの多い問題について，より効果的に学習者に意識付けさせる方法の検討や，修正方法の最適化を行う．また，自学自習用の品質評価ツールを実装して，学習者に平時から品質を意識したプログラミングを行える環境を提供する．

参考文献

[1] ISO/IEC 9126-1:2001, http://www.iso.org/iso/
[2] M.Fowler, K.Beck, J.Brant, W.Opdyke, D.Roberts, Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.
[3] Stephen H.Kan, Metrics and Models in Software Quality Engineering (2nd Edition), Addison-Wesley, 2002.
[4] 若林智徳，松浦佐江子，“オブジェクト指向プログラミング初学者のためのソースコード品質評価分析”，信学技報，vol.111, no.282, KBSE2011-38, pp.13-18, 2011.
[5] Web 教科書システム, http://www.sayo.se.shibaura-it.ac.jp/incusphere/webStudy.html