



Title	ソースコードに対する適用可能な修正手順を可視化するリファクタリング支援手法の提案
Author(s)	譜久島, 亮; 吉田, 則裕; 松下, 誠 他
Citation	
Version Type	VoR
URL	<a href="https://hdl.handle.net/11094/50640">https://hdl.handle.net/11094/50640</a>
rights	ここに掲載した著作物の利用に関する注意 本著作物の著作権は情報処理学会に帰属します。本著作物は著作権者である情報処理学会の許可のもとに掲載するものです。ご利用に当たっては「著作権法」ならびに「情報処理学会倫理綱領」に従うことをお願いいたします。
Note	

*The University of Osaka Institutional Knowledge Archive : OUKA*

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

# ソースコードに対する適用可能な修正手順を可視化するリファクタリング支援手法の提案

譜 久 島 亮<sup>†</sup> 吉 田 則 裕<sup>†</sup>  
松 下 誠<sup>†</sup> 井 上 克 郎<sup>†</sup>

リファクタリングを行う際に必要となるソースコードの修正作業は複雑になりやすい。本研究では、各リファクタリングパターンを適用する際に適用可能な修正手順の導出・可視化手法を提案する。

## 1. リファクタリング支援ツールの問題点

リファクタリングとは、ソースコードの外部的振舞いを保ったまま内部構造を改善する作業である<sup>1)</sup>。文献1)は、“メソッドの移動”などの典型的なリファクタリングをリファクタリングパターン(以降、パターン)としてまとめている。

パターンに基づく修正作業には、ソースコードや開発者の意図により、適用可能な修正手順が複数存在する。その修正作業は、ソースコードを頂点、修正手順を構成する各ステップ(以降、ステップ)を有向辺とする木構造で表現できる。木の根はリファクタリング開始時のソースコード、葉はリファクタリング完了後のソースコードを指し、道は修正手順の1つを表す。図1は、“メソッドの移動”完了後のソースコード例であり、図2は図1(a)の“メソッドの移動”に伴う修正作業を表す木である。図2に示す5つの道のうち2つが、それぞれ図1(b)、図1(c)を得るために手順である。

しかし、既存のリファクタリング支援ツールでは適用可能な修正作業を表す木を提示するものが確認できていない。よって、開発者は適用可能な修正手順を理解しないまま修正作業を行う可能性がある。その結果、“目的のソースコードを得ることができない”、“ステップの適用や適用の取り消しを繰り返すことで作業時間が大きくなる”という問題が生じる可能性がある。

## 2. 提案手法

リファクタリングを行う際に適用可能な修正手順を提示する手法を提案する。まず各パターンにおいて最低限必要なステップを適用し、発生した意味解析上のエラー(以降、エラー)をコンパイラから取得する。次

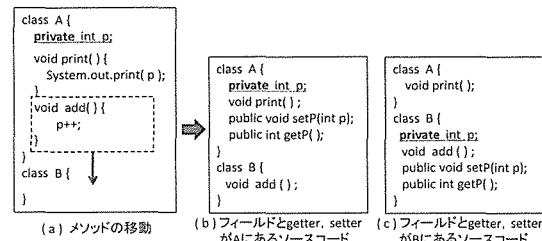


図1 “メソッドの移動”リファクタリングによる完了後のソースコード例(完了後のソースコードに含まれる関数本体は省略)

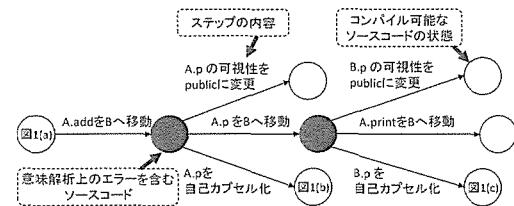


図2 図1(a)の“メソッドの移動”に対応する修正作業の木

にエラーを解決するステップを導出・適用する。図2の例では、まず最低限必要なステップである“A.addをBへ移動”を適用し、エラーを取得する。続いてエラーから導出されるステップである“修飾子の変更”や“移動”、“フィールドの自己カプセル化”を適用する。このようにエラーが全て解決されるまでステップを導出・適用し、修正作業を表す木を提示する。

## 3. 今後の課題

本手法はEclipse Plug-inとして実装する。エラー取得やソースコードの自動変更はEclipseの機能を利用する。加えて今後、修正手順の妥当性を検討する。

## 参考文献

- 1) Fowler, M.: Refactoring: improving the design of existing code, Addison Wesley (1999).

<sup>†</sup> 大阪大学 大学院情報科学研究科

Graduate School of Information Science and Technology, Osaka University