4R−5

リファクタリングによるソフトウェア品質改善と品質劣化の予防
(2) リファクタリング支援ツール Refactoring Assistant

本間 昭次　片岡 欣夫　深谷 哲司
株式会社 東芝　研究開発センター　システム技術ラボラトリー
{ akitsugu.homma | yoshio.kataoka | tetsuji.fukaya }@toshiba.co.jp

1 はじめに

リファクタリングはプログラムの保守性向上や品質向上に有効であると言われている．実際に一部の熟練したプログラマは定常的に行ってきた．しかし，その一方で現場では広く受け入れられていないのが実情である．その原因として次のようなリファクタリングに対する障壁が考えられる．

・いつ，どのようにリファクタリングしたらよいか分からない
・リファクタリングをする時間がない
・リファクタリングによる不具合の作り込みが不安である

リファクタリングを定常的に行い，品質の改善や劣化防止をするためには，リファクタリングの障壁を低くする必要がある．その手段の一つとして，リファクタリングを支援するツールが[1]を始めいくつか提案されているが，いずれもリファクタリングのプロセスを部分的に支援するに過ぎない．我々は，リファクタリングのプロセスをトータルに支援するリファクタリング支援ツール Refactoring Assistant を提案する．

2 Refactoring Assistant の全体構想

Refactoring Assistant が目指す支援を，我々が提案する，三つのサブプロセスから構成されるA（アルファ）プロセス[2]に沿って説明する．

a. リファクタリング候補同定
・プログラムを検査し，リファクタリングを行うべき箇所の指摘や改善方法の提案を行う（開発者レベル）．
・プログラム構造を視覚化し，品質劣化の原因を分析する環境を提供する（分析者レベル）．
・改善のコストと効果を提示する（管理者レベル）．

b. リファクタリング適用
リファクタリングの実施を自動化し，手間を減らすとともに，変更漏れを防止する．（開発者レベル）．

c. リファクタリング結果検証
・リファクタリング結果の検証を行い，その前後で動作が変わっていないことを保証する（開発者レベル）．
・プログラムの品質を定量的に測定し，リファクタリングによって品質が改善されていることを確認する（分析者，管理者レベル）．

現在，候補同定の支援系が完成しており，実施の支援系も一部実装が完了している．以下では候補同定の支援について述べる．

3 リファクタリング候補同定支援

リファクタリング候補を同定するために必要な各ステップについて，課題とその支援策とを述べる．

3.1 Bad-smell[3]の発見

Bad-smell とは，リファクタリングの必要性を示唆するコーディングの特徴と定義することができる．

一般に，Bad-smell はソースコードのレビュー等により第三者の目に触れられなければ発見されず，放置されたままとなる．しかし，レビューするにしても時間がかかり，膨大なプログラムを全てチェックするのは事実上不可能である．

このため，プログラムを解析し，自動的に Bad-smell を検出する支援が有効である．これにより実用的な時間で広い範囲にわたる Bad-smell の発見が可能となる．

3.2 Bad-smell の原因分析

Bad-smell はその意味を考えずに表面的に消すこともできてしまう．しかし，それでは問題を隠蔽するだけで改善にはならない．原因を分析し，問題の本質を理解するためには，指摘箇所だけでなくその周辺箇所も含めた実装や設計の理解も必要になるため難しい．

そこでユーザに対し，検出された Bad-smell とその理由，および指摘箇所のコードを提示し，原因を問い掛けるという支援が考えられる．それによりユーザは指摘された理由を考え，問題を自覚することが可能となる．

また，ユーザに対してプログラム構造を視覚化するブラウザを提供することも有効な支援である．それにより原因分析に必要なプログラムの設計や実装の理解が容易になる．

3.3 リファクタリング方法の決定

リファクタリングのコストと効果の定量化が難しく，限られた開発コストの中でリファクタリングを効率的に選択することができない．また，実施するにしても，その方法についての知識が不足しているため，安全に実施することができない．

そこで検出された Bad-smell を集計し，その結果に基づき改善コストと効果の予測ができるようにする．この結果対処する Bad-smell を効率的に決定することができる．

また，各 Bad-smell から導かれる原因別にリファクタリング方法の候補を提示することにより，ユーザに対処方法を知らせるという支援も有効である．

Software Quality Improvement and Deterioration Prevention using Refactoring − Refactoring Tool "Refactoring Assistant"
Akitsugu HOMMA, Josh KATAOKA, and Tetsuji FUKAYA, Systems Engineering Laboratories, Corporate Research & Development Center, TOSHIBA Corporation

4 機能

上記要件を満たすために，Refactoring Assistant は以下の機能を提供する(図1)．

・解析機能(プリプロセス，字句解析，構文解析)

ソースファイルを解析し，構文木とクロスリファレンスを作成する．

・Bad-smell 検出機能

メトリクスを計測し，閾値を超えるメトリクス値を検出する．これにより Bad-smell の発見を支援する．対象プロジェクトに合わせ，計測するメトリクスと閾値を指定することができる．

・Bad-smell の情報提示機能

検出された Bad-smell に対する説明，原因を問い掛ける質問，リファクタリング方法の説明，該当箇所のソースコード，および関連する Bad-smell を提示する．これによりユーザが Bad-smell の意味を正しく理解し，リファクタリングを正しく行うことができる．また，これらの情報は後段のリファクタリング適用支援でも利用可能である．

・クロスリファレンス表示，シンボル検索機能

クロスリファレンス(コールグラフ，コールドグラフ，クラス図)表示機能，シンボルの使用箇所や別名の検索機能を提供する．クロスリファレンスでは別名を考慮し，例えば仮想関数のような動的に決定する呼び出しでは，呼び出される可能性のある候補をすべて表示する．これにより，プログラムの理解が容易になり，Bad-smell の原因分析が可能となる．

・Bad-smell のファイル出力機能

検出された Bad-smell と指摘箇所を統計ツールで利用できるようにファイル出力する．これにより Bad-smell の集計が容易となり，改善のコストと効果の予測が可能となる．

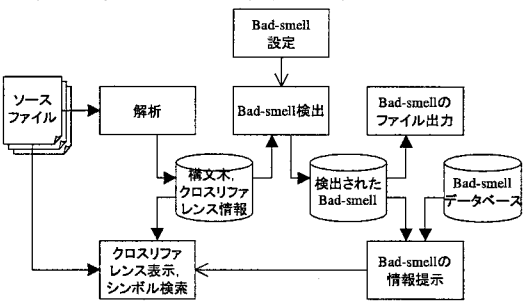

図 1　Bad-smell に関する情報の提示

5 評価

Refactoring Assistant が実用的な時間でリファクタリング候補の同定が可能であることを示すための実証実験について述べる．

5.1 実験方法

Bad-smell の検出を，Refactoring Assistant を使用した場合と使用しない場合とで比較した(表1,2)．

表1 実行環境

CPU	Pentium III 700MHz
Memory	512Mbyte
OS	Windows NT 4.0

表2 対象プログラム

記述言語	C++
ソースファイル数	21
ヘッダファイル数	30
総行数	4896
クラス数	55

5.2 結果

Bad-smell 検出にかかった時間を表3に示す．Refactoring Assistant では，2 回目以降の解析では変更されたソースファイルを解析すればよく，解析時間はさらに短縮できる．検出した Bad-smell を7種類に分類した結果を表4に示す．

Refactoring Assistant を使用することにより，プログラム全体を対象とした Bad-smell の検出を実用的な時間内に行うことができた．また，重複コードや使われていないコードのような複数の箇所を見なければ検出できない Bad-smell も，容易に検出することができた．

表3 Bad-smell 検出時間

Refactoring Assistant		不使用
解析	44 s	16 h
Bad-smell 取得	4 s	
合計	48 s	

表4 検出された Bad-smell の分類結果

種類	Refactoring Assistant	不使用
重複コード	13 件	0 件
分割した方がよいコード	36 件	6 件
まとめた方がよいコード	4 件	6 件
役割の少ないコード	102 件	10 件
使われていないコード	9 件	0 件
不適切なスコープ	16 件	0 件
その他	29 件	9 件
合計	209 件	31 件

6 おわりに

リファクタリングを容易に行うことを支援するツール Refactoring Assistant のうち，主にリファクタリング候補同定の支援について述べた．

今後は，検出された Bad-smell について，その有効性の評価を行う予定である．また，現在リファクタリング適用支援についても一部実装されているが，今後は結果検証も含め，さらに高度な支援を行っていく．

参考文献

[1] 原 智明 他，オブジェクト指向プログラム構造に基づいた情報の提示によるリファクタリング支援システム，情報処理学会第 62 回全国大会，2001

[2] 片岡 欣夫 他，リファクタリングによるソフトウェア品質改善と品質劣化の予防 (1) リファクタリングプロセスの定義，情報処理学会第 63 回全国大会，2001

[3] M. Fowler. Refactoring: Improving the Design of Existing Code. Addison-Wesley, 1999.