R-1

パターンの役割違反に着目したコード品質の可視化手法の提案

橋浦研究室　　1175010 飯山 隆章　　1175236 山崎 貴弘

1. はじめに

今日のソフトウェア開発において，GoF のデザインパターン[1](以下，デザインパターン)が広く用いられている．一方で，デザインパターンを正しく使用できなかった場合に元のコードよりコードの品質が低下を引き起こすという問題がある[2].

2. パターンエラー

本研究では前述のようにデザインパターンを利用したことによって引きおこされる問題をパターンエラーと呼ぶ.

開発者が自らのコードにデザインパターンを組み込むためには，クラスごとに満たすべき条件（以下ロール）を正しく認識し，パターンに含まれるクラスと自らのコードに存在するクラスをロールによって対応付ける必要がある．ロールには，それを構成する複数の要素(以下，特徴と呼ぶ)が存在する．デザインパターンの組み込みにあたっては，ロールが割り当てられたクラスに対して，これらの特徴を過不足なく組み込む必要がある．もし，組み込む先のクラスが自らのコードに存在しない場合には，新たにクラスを追加したり，既存のクラスを分割するなどして，ロールの対応かつそのロールの特徴を組み込む必要も生じる．パターンエラーの発生を防ぐためには，このような作業を確実に行なう必要がある．

3. 研究目的

本研究は，前述のパターンエラーの検出をツールによって自動的に行うことで，開発者自らがパターンエラーを除去できる手法を提案するものである．

4. 提案手法

本研究ではデザインパターンの初学者であっても，適切な修正ができるようにするために，初学者にパターンエラーを引き起こしている箇所を提示する必要がある．このことを踏まえ，本研究ではパターンエラーの原因となるロールを満たしていない特徴を自動で検出し，その原因と修正方法を具体的に指示することで初学者に対してパターンエラーの修正を促すツールを作成した.

ツールを実現するためには，まず，パターンに含まれるロールの特徴を明確にする必要がある．本研究で扱う特徴は，クラス図に用いられる 2 つ以上のクラス間の関連によって定義されるもので，文献[3]のクラス間の関連をもとに継承，集約，委譲，オーバーライド，ラッパー，インスタンス化の 6 つに着目して定義を行った.

5. 実現手法

ロールの特徴を検知するために，まず始めに Eclipse に含まれる抽象構文木（ASTVisitor）を用いて，コードの静的解析を行う．次に，解析の結果と文献[1]で定義されているパターンのロールを照らし合わせる．最後に，パターンのロールの特徴について，その過不足を調べ，その結果を可視化する (図 1).

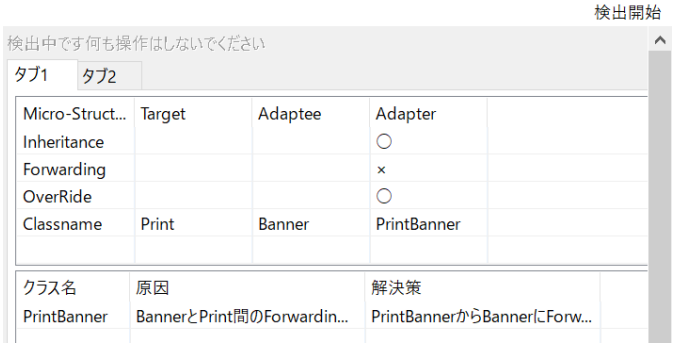

図 1 ツールの実行画面

6. 実験

本手法の可視化手法の有効性を確認するためにオブジェクト指向プログラミングに理解がある日本工業大学の 3，4 年生の 9 名に対する実験を行なった.

実験の概要は被験者にパターンエラーが 1 つだけ含まれたコードを提示し，その修正を行なわせるものである．実験に用いたデザインパターンは，Factory Method，Composite，Adapter の 3 つとし， 各パターンに挿入したパターンエラーについては表 1 に示した通りである.

表 1 挿入したパターンエラー

パターンの種類	ツールの有無	パターンエラー
Factory Method	あり	インスタンス化
	なし	継承
Composite	あり	集約
	なし	継承
Adapter	あり	継承
	なし	委譲

パターンエラーについては，あるエラーを混入すると他のエラーも混入されてしまうなどの相互作用が発生する可能性がある．例えば，継承に関するパターンエラーを挿入した場合，オーバーライドのパターンエラーも同時に発生する．これは，オーバーライドを行

う前提として，継承が行われている必要があることに起因している．今回の実験ではこのようにパターンエラーの相互作用で発生する複数のエラーを単一のものとして取り扱う．具体的には，1 箇所の誤りを修正することで複数のパターンエラーが修正されるように問題を作成している．

7. 評価方法

被験者のコード内に存在する特徴に対して，パターンエラーの数を計測し評価基準とした．各パターンの特徴数は，それぞれ Factory Method は 6，Composite は 5，Adapter は 3 である．

8. 結果と考察

ツールの有無によるパターンエラーの変化を表 2 に示す．

表 2 ツールの有無によるパターンエラーの変化

#		エラーあり	エラーなし	合計
1	ツールあり	20	106	126
2	ツールなし	44	82	126
3	合計	64	188	252

表 2 のツール有無によるエラーありの数に着目すると，結果に大きな差が確認できる．これは，ツールがクラスとロールを対応付けることにより，パターンエラーの原因と修正方法が適切に提示したことで被験者の修正を促すことができたためであると考えられる．表 2 のデータに対して χ^2 検定を行ったところ，有意差が確認された $(\alpha = 0{,}05,\ p = 0.0005141)$. これにより，ツールを用いることでエラーの低減することができる可能性が示唆された．

さらに，個々のパターンのパターンエラーの修正について表 3,4,5 に示す. これらに χ^2 検定を行った結果，Factory Method (表 3)の結果にのみ有意差が認められた $(\alpha = 0.05,\ p = 0.00062137)$. 表 2 に対する結果と同様に，表 3 のツールの有無によるエラーありのエラー数に着目すると，ツールを用いることで，個々のパターンに対してもパターンエラーの修正を適切に促していることが確認できる．また，継承のパターンエラーのエラー数に着目すると，ツールなしに比べ，ツールを用いたほうがより修正を行えていることが確認できる．これは，継承のパターンエラーを修正できない被験者の特徴として，継承の宣言が必要なクラスを見つけることができない人が多く見られた一方，ツールが継承の宣言が必要なクラスを可視化したことで，被験者に修正を適切に促すことができたためであると考えられる．

しかしながら，集約のパターンエラーの修正結果に着目すると，他のパターンの場合より修正できた割合が低かった．これは，ツールの可視化の限界であると考えられる．現状のツールは，パターンエラーを起こしているクラスを提示することはできるが,例えば「クラス内のメンバー変数をあるメソッド内の変数に代入せよ」といったクラス内に対する具体的な修正方法の提示ができない．このためエラーを修正する場合，被験者は具体的な修正方法を自ら考える必要が生じる．このことにより，継承も対してはクラスが理解できれば修正を実施できたが，集約に対しては修正できない結果となった．

表 3 Factory Method パターンの実験結果

#		エラーあり	エラーなし	合計
1	ツールあり	5	49	54
2	ツールなし	20	34	54
3	合計	25	83	108

表 4 Composite パターンの実験結果

#		エラーあり	エラーなし	合計
1	ツールあり	10	35	45
2	ツールなし	13	32	45
3	合計	23	67	90

表 5 Adapter パターンの実験結果

#		エラーあり	エラーなし	合計
1	ツールあり	5	22	27
2	ツールなし	11	16	27
3	合計	16	38	54

9. 結論

ツールを用いることで，パターンエラーの修正に有意であることが確認できた．これにより，デザインパターンを正しく使用しないと元のコードよりコードの品質が低下する問題に対し，部分的にではあるが有効な方法を提案できた．

今後の課題として,パターンエラーを起こしているクラスの具体的な修正方法を可視化することで集約のパターンエラーの修正率の向上を目指す．

文　献

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "オブジェクト指向における再利用のためのデザインパターン," ソフトバンク, 1994 年 10 月.

[2] Naouel Moha, Duc-loc Huynh and Yann-Gaël Guéhéneuc, "A Taxonomy and a First Study of Design Pattern Defects," Proc. of the STEP International Workshop on Design Pattern Theory and Practice (IWDPTP05), pp. 225-229, Jan. 2005.

[3] 結城 浩,”java 言語で学ぶデザインパターン入門,” SB クリエイティブ株式会社，2019 年 5 月.