6A-1

設計ルール逸脱箇所検出による設計改善手法

山﨑　愛†　　大場　勝†　　河原　敏宏†　　岡本　渉†
株式会社東芝　ソフトウェア技術センター†

1. はじめに

現在のソフトウェア開発は、新規開発よりも既存資産に追加・変更を行う派生開発が多くを占める。開発当初は、設計方針や設計ルールを設定し、それに従って設計をすることにより、保守性の優れた設計を実現する。しかし、派生開発を重ねると、場当たり的な修正により設計ルールから逸脱した実装が発生し、その結果保守性が低下するといった問題がある。低下した保守性を回復もしくは開発当初の保守性を維持するためには、既存資産の設計ルールに従って修正を行う必要がある。

設計ルールから逸脱する原因として、設計ルールを設定した設計者と派生開発による追加・修正を行う開発者が異なるため、既存資産にどのような設計ルールが適用されているかを、開発者が把握できないことが挙げられる。既存資産の設計書には設計結果だけを記載される場合があり、設計時に設けたルールが必ずしも記載されているとは限らない。また、人手によりソースコードを読み解き、適用されている設計ルールを把握するには多大なコストがかかる。そこでソースコードから、当初の設計において適用されていたと考えられる設計ルールを推測し、その上で、設計ルールからの逸脱箇所を検出する手法を提案する。

2. 設計ルール

本手法で、対象とする設計ルールについて述べる。設計ルールは、設計パターンと制約から構成される。設計パターンとは、構造の特徴を体系化したものであり、例として、GoF のデザインパターン[1]が挙げられる。その設計パターンに対し、遵守すべき制約を述べたものが設計ルールである。従って、設計ルールは「“設計パターン”において“制約を遵守していること”」といった形式で表現可能である。

2.1　設計ルールの例

図 1 を用いて設計ルールの例を示す。次の 3 つの条件を満たす構造を、ここでは同格モジュールパターンと呼ぶ。

(1)依存元モジュール(以下、親モジュール)は 3 つ以上の依存先モジュール(以下、子モジュール)を持つ。
(2)子モジュール同士は独立している。
(3)親モジュールから呼び出されている子モジュールの関数は、親モジュールの関数にのみ呼び出される。

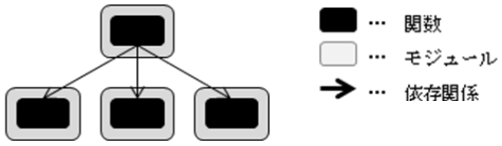

図 1　同格モジュールの設計パターン例

同格モジュールパターンにおいて想定される制約には、”親子関係のないモジュール(以下、第三者モジュール)は、子モジュールに依存しないこと”等が考えられる。従って、設計ルールとしては、「同格モジュールパターンにおいて、第三者モジュールは、子モジュールに依存しないこと」が一例として挙げられる。

3. 設計ルール逸脱箇所検出の手順

本手法では、以下の手順により設計ルールからの逸脱箇所を検出する。

手順 1.　設計ルール適用箇所の候補を抽出
手順 2.　設計ルール適用の有無を判定
手順 3.　設計ルールの逸脱箇所を提示

具体的には、まず、設計ルールとして、一般的に何があり得るかを予め列挙しておく。それぞれの設計ルールの設計パターンを対象のソースコードに、順に当てはめて比較し、設計パターンに合致している箇所を抽出する(手順 1)。これにより、適用されている設計ルールが事前に明らかでない場合であっても、適用されている可能性の高い設計ルールとその適用箇所の候補を抽出することが可能である。しかし、手順 1 で抽出した設計ルール適用箇所の候補には、偶然設計パターンと一致した構造として検出された

Detecting Design Rule Violations
†Ai Yamasaki, Masaru Ohba, Toshihiro Kawahara, Wataru Okamoto
†Corporate Software Engineering Center, Toshiba Corporation

が、制約を満たしておらず、設計ルールは適用されていないものが含まれている可能性が考えられる。そこで、閾値を設定し、設計ルール適用箇所の候補から、制約を満たしているモジュール数や関係数が閾値以上である場合、設計ルールを適用されていると判定する(手順 2)。ルールが適用されている場合、適用箇所の中で制約を逸脱している箇所を提示する。(手順 3)

3.1 設計ルール逸脱箇所検出の例

前章で述べた「同格モジュールパターンにおいて、第三者モジュールは、子モジュールに依存しないこと」の設計ルール抽出例を示す。
手順 1 として、ソースコードから、同格モジュールパターンの条件をすべて満たす箇所を抽出する。図 2 は、ソースコードから抽出したソフトウェア構造と同格モジュールパターンの構造が一致した範囲を示す。

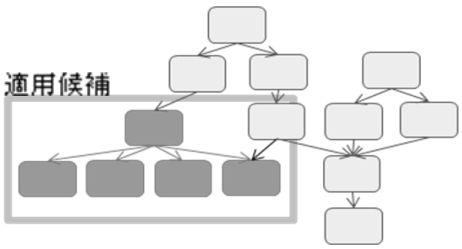

図 2 同格モジュールパターンと合致したソフトウェア構造

次に、適用箇所の候補から適用されている可能性が高い箇所を絞り込む。本例では設計ルールの適否を判断する閾値を 50%とする。図 3 では 4 つの子モジュール(B,C,D,E)の中で E モジュールのみ第三者モジュールが依存している。制約を遵守しているモジュールが過半数のため本ルールは適用されていると考えられる。

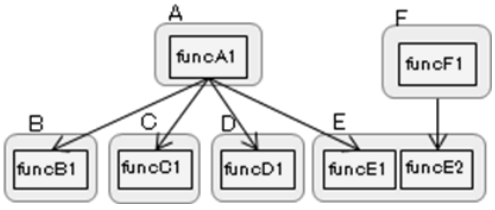

図 3 設計ルール適用箇所の候補

この場合、F モジュールから E モジュールへの依存が逸脱箇所となり、E モジュールには A モジュールからのみ依存可能であることを提示する。

4. 設計改善手法の適用事例

3 章で述べた、設計ルール逸脱箇所の検出手法を、ある組み込みソフトウェアに適用した。本事例では、設計ルール「同格モジュールパターンにおいて、第三者モジュールは、子モジュールに依存しないこと」に対して検出を行い、適用されている範囲及び逸脱箇所を検出できることを確認した。図 4(上)では、最下段に並列しているモジュール群が子モジュールであり、依存元の太枠のモジュールが親モジュールである。第三者モジュールが依存している子モジュールが存在するため、その依存関係を設計ルール逸脱箇所とする。このように設計ルールとその逸脱箇所を明示することにより、保守性低下の原因を特定できる。

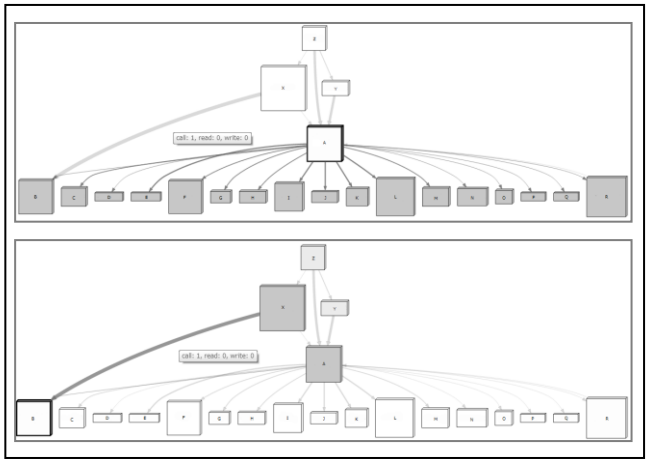

図 4 検出した設計ルールの適用範囲(上)および逸脱箇所の強調(下)

5. おわりに

ソフトウェア開発では、保守性の優れた設計が求められるが、派生開発により適切でない修正が繰り返され、保守性が低下する場合がある。本稿では、設計ルールが設計書に充分に記載されていない場合であっても、設計意図を推測し、設計意図から逸脱している箇所を指摘する手法を提案した。これにより、人手によるコストをかけず、設計ルールに沿った状態に保守性を回復・維持することが可能となる。今後は、検出可能な設計ルールの拡充を行う。

参考文献

[1] Erich Gamma, Ralph Johnson, Richard Helm, John Vlissides, ”Design Patterns Elements of Reusable Object-Oriented Software”, 1995.