コード検査ツールのためのテストケースの自動生成に関する研究

2010SE164 小椋正勝
指導教員：野呂昌満

1　はじめに

コードインスペクションツールは，ソースコードを静的に検査し潜在的な欠陥を検出するツールである．本研究室では Java ソースコードを対象とするコードインスペクションツール (以下，JCI) が開発されている [1]．JCI の検査の品質はテストをおこなうことで管理している．現在 JCI のテストケースは自然言語で書かれた仕様書を基に手作業で作成している．JCI の検査の品質を確認するには，多種多様なテストケースが必要であり，テストケースの作成，修正において開発コストがかかる．

JCI は加藤によって新たなアーキテクチャが提案されている [2]．提案されたアーキテクチャによって，JCI の検査仕様は状態遷移モデルで記述することが可能であるとわかった．状態遷移モデルで記述された検査仕様からテストケースを自動生成可能であると考えられ，その方法が提案された．一方，提案された JCI のテストケースの自動生成系は実現されておらず，テスト設計の省力化は課題である．

本研究の目的は，テストケース自動生成系の実現によるテストプロセスの半自動化である．検査仕様記述からテストケースを生成する手法を提案し，自動生成系を実現する．テストケースの自動生成系によって必要十分なテストケースを生成し，テストをおこなう．結果，JCI の検査におけるテスト漏れを排除し，実現した検査の品質が確認できる．

本研究では，テストケースの自動生成を構文木を辿る問題として再定義する．状態遷移モデルを用いて検査仕様記述を記述し，状態遷移モデルのアクションと出力すべきコード片の対応付けを定義する．状態遷移のアクションに対応したコード片を出力することで，テストケースの自動生成を実現する．

2　JCI の概要

JCI は Java ソースコードの中から不具合を起こす可能性のある箇所を検出，指摘するツールである．JCI の検査項目では 36 項目の検査項目が実現されており，検査仕様記述には自然言語と警告例が用いられている．より容易にカスタマイズ可能なコード検査ツールの実現を目的に，アスペクト指向技術を適用した新たな JCI のアーキテクチャが提案されている [2]．新たな JCI における検査仕様記述には，状態遷移モデルが用いられている．

3　検査仕様記述の定義

先行研究 [2] で提示された検査仕様記述は，検査対象を含む要素の出現を示すイベントが 1 つしかない．検査対象を含む要素が複数現れた場合，検査を正しくおこなうことができない可能性が明らかになった．検査仕様記述の修正をおこなった結果，新たに別の状態遷移モデルのパターンを用いて検査仕様を記述することができた．検査仕様のパターンを図 1，図 2 で示す．

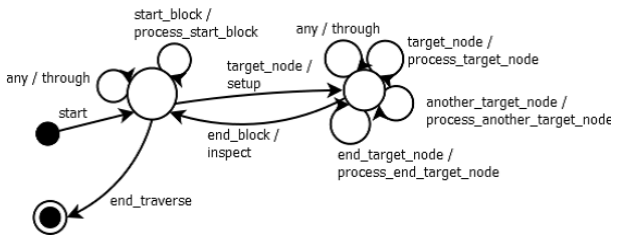

図 1　検査仕様のパターン

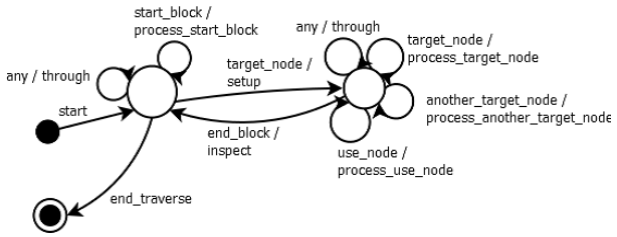

図 2　検査仕様のパターン

図 の　target_node，use_node，another_target_node，end_target_node は検査対象に関わる要素の出現や終わり，start_block，end_block はブロック文，any は無視するイベント，end_traverse は検査の終了をあらわす．

JCI がおこなう検査は，検査内容を分類すると抽象構文木の構造に関する検査，変数の宣言と使用に関する検査，制御フローに関する検査に分けられる．抽象構文木の構造に関する検査は図 1，変数の宣言と使用に関する検査は図 2 のパターンで記述できる．制御フローに関する検査は，抽象構文木の構造に関する検査と同様のパターンで記述できる．

4　テストケース自動生成の枠組み

本研究では，状態遷移モデルのパターンに着目し，遷移のアクションを利用する．本研究のコード片生成は，UNPARSING SCHEME[1] を用いた．UNPARSING SCHEME は遷移名に対応付けられるコード片の定義である．UNPARSING SCHEME を利用し，アクションとコード片の対応付けによってテストケースを自動生成する．テストケースの生成では，検査仕様の全ての遷移を一度通る経路を利用する．遷移の優先順位は，他の状態への遷移より自己遷移を優先する．一つの状態に対して複数の遷移が存在する場合，遷移順を記述する必要がある．図 3

を用いてテストケース生成の流れを説明する．

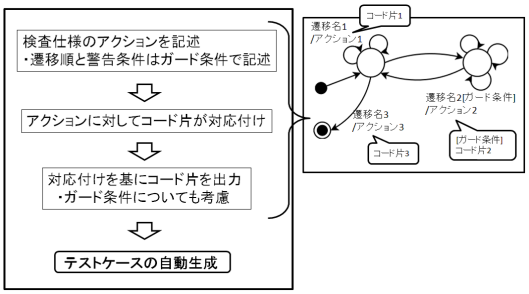

図 3 テストケース生成の流れ

検査仕様のアクションを記述する．警告条件と複数の遷移が存在する場合の遷移順は，イベントのガード条件として記述する．アクションと UNPARSING SCHEME が対応付けされる．対応付けを基に，ガード条件をふまえてコード片を出力する．結果，テストケースが自動生成できる．

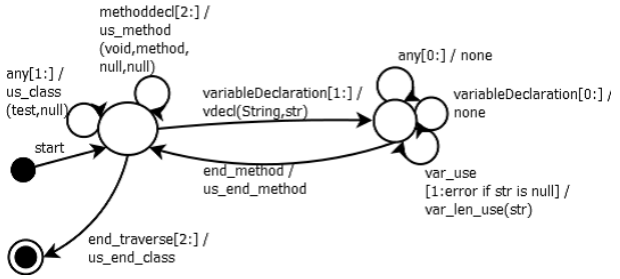

図 4 アクションを記述した検査仕様

図 4 は，ヌル値の参照を検出する検査仕様を状態遷移モデルで表し，アクションを記述したものである．この仕様から，テストケース自動生成の枠組みで定義した遷移順に従い，1 つ目の状態の any，methoddecl，variablDeclaration，var_use，end_method，end_traverse の順の経路を得る．遷移順 0 のイベントは経路に含まない．この経路のアクションを利用し，テストケースを生成する．

クラスの宣言はテストケースに必要なイベントだが，検査仕様上では無視するイベントとなっている．よって，同じく無視するイベントを表す any のイベントのアクションで，クラスの宣言のコード片を出力する．アクションには，メソッド名や変数名といった情報を追記する．結果，アクションと UNPARSING SCHEME の対応関係を基に，経路に従って us_class(test,null) に対して “ class test { “ , us_method(void,method,null,null) に対して“ public void method(){ “ , vdecl(String,str) に対して “ String str; “ , var_len_use(str) に対して“ str.length();“ , us_end_method に対して “ } “ , us_end_class に対して “ } “ の順でコード片を出力する．また, “ length.str(); “ のコード片前に，ガード条件で記述した変数 str がヌル値の場合という条件を反映し, “ str = null; “ を出力する．結果，以下のようなテストケースを得られる．

得られるテストケースの例

```
class test {
   public void method(){
      String str;
      str = null;
      str.length();
   }
}
```

5 考察

検査仕様記述のアクションと出力したいコード片は一意に対応付けられた．イベントに適切なアクションを設定し，対応したコード片を出力した．結果，検査仕様に対し適切なテストケースが一意に生成できたので，アクションとコード片の対応付けは妥当であると考える．

検査項目に関わらない要素は検査仕様で着目する必要がない．検査仕様記述の修正をおこない，検査仕様で着目すべきイベントとその他のイベントを整理した．結果，検査項目に関わらない要素は全て検査仕様で着目しないイベントとして記述することができた．これにより，パターンによって全ての検査仕様を状態遷移モデルで記述することができたので，イベントの区別は妥当であると考える．

コード片の生成は UNPARSING SCHEME を用いた．仕様上では現れない情報をアクションに記述し，対応付けたコード片を出力することで，テストケースを生成した．しかし本研究で提案した手法は，それぞれの検査仕様に対し個別の UNPARSING SCHEME を定義する必要がある．仕様の数だけ定義付けが必要であり，標準化ができていない．この問題について，複数の構文要素を一つのイベントとしてまとめることで，状態遷移モデルのイベントを標準化できる．着目するイベントを標準化することで，UNPARSING SCHEME を標準化できると考える．

6 おわりに

本研究では，状態遷移モデルによる検査仕様記述のアクションとコード片を対応付け，対応付けによるテストケース自動生成の方法を提案した．対応付けを用いて検査仕様からテストケースを生成できることを確認した．生成したテストケースを基にテストをおこない，実現した検査の品質が確認できるようになった．今後の課題として，UNPARSING SCHEME の標準化と，対応付けを利用したテストケース生成器の設計が挙げられる．

参考文献

[1] M. Noro, and A. Sawada, “ Aspect-Oriented Software Architecture for CDI Tools: toward PLSE Construction, ” *Technical Report of the Nanzan University Academic Society Information Sciences and Engineering*, NANZAN-TR-2012-02, 2012.

[2] 加藤遼介, “ コード検査ツール開発における検査仕様記述の提案とテストケースの自動生成に関する研究, ” 南山大学大学院数理情報研究科，2013.