抽象構文木に基づくテストケース生成支援ツールに関する研究
– 表を用いた仕様記述 –

2005MT011 藤井 祐一郎　　2005MT096 佐藤 彰洋　　2005MT126 山田 大介
指導教員　野呂 昌満

1　はじめに

コーディングを支援するツールとして Code Inspection(以下 CDI）ツールが存在する．CDI ツールとは，ソースコード中から誤りの可能性のある箇所を検出する静的解析ツールであり，機能テストには，テストデータとしてソースコードを用いる．ソースコードは CDI ツールが検査する内容毎に複数作成する必要があり，多くの労力を要する．一方で，テストデータとなるソースコード毎の差異は一部の構文要素であり，差異となる構文要素を変更することで新たなテストデータを作成可能であり，この変更作業を自動化することで，テストデータ作成の省力化が可能となる．既存のソースコードを変更するツール [1] では，ソースコードを実行した際の振る舞いを変更することを目的としておらず，振る舞いの変更に必要が無い構文要素を指定して変更することができず，テストデータの生成には不十分である．よって，特定の構文要素を指定して変更を行うことが可能なテストデータ生成支援ツールが求められる．構文要素間の位置関係を記述することで，特定の構文要素を指定可能な文献 [2] の言語が存在する．

問題として，構文要素の指定方法に前述の言語を用いた場合，変更に必要な情報の記述量が多く，指定する情報の入力に手間が掛かる．また，言語の文法をツールの使用者が習得しなくてはならず，テストデータ生成支援ツールへの容易な入力手段とはいえない．

本研究の目的は，CDI ツールのテストデータ生成支援ツールに対する容易な入力手法の提案である．我々は入力手法として，表を用いた入力形式を提案することで，言語の文法を習得することなく本ツールのユーザが容易に入力できる記法を提案する．

本研究は以下のように進めた．

1. CDI ツールの検査毎のテストデータの差異の分析
2. 表に入力する情報の整理
3. 容易な入力のテストデータ生成支援ツールの提案
4. ツールの入力形式の設計
5. 考察

2　研究背景

2.1　背景技術

本節では，特定の構文要素間の位置関係を指定する既存の技術のひとつである metal[2] について述べる．

metal とは静的解析ツールで検査する規則を定義する言語である．metal は誤りとなる可能性がある構文要素の並びを記述することで，ソースコード中から記述と一致する箇所を発見することを目的としている．

metal では，状態，状態遷移を行なう構文要素の並び，その構文要素が存在した場合の振る舞いを記述する．metal を用いて CDI ツールの検査内容を記述した例を次に示す．

二重 While 文があった場合に警告する検査

```
start: {  WhileStatement } ==> InWhile;
InWhile: { WhileStatement } ==> Stop,
{ err("二重 While 文"); };
```

この例では，start の状態から開始し，ソースコード中に While 文があった場合に，InWhile という状態に遷移する．次に，発見した While 文の中を探索したときに，再度 While 文があった場合に，二重の While 文を発見したとして警告を行う．

記述した構文要素が存在した場合に，その振る舞いを記述するという手法を参考にする．

2.2　CDI ツールに対するテストデータ

本節では表に記述するテストデータの変更に必要な情報の整理のために，テストデータの特徴，新たなテストデータ作成に必要な情報を述べる．

2.2.1　特徴

CDI ツールの検査内容毎のテストデータを分析した結果，テストデータ毎の差異の一例として次のようなものがある．

- 制御文の種類の違い
- 数値リテラルの違い
- 式文の有無

これらの差異を整理した結果，CDI ツールに対するテストデータを作成するには，「構文要素の有無の変更」，「構文要素の種類の変更」という変更が必要になる．よって，テストデータに対して，「構文要素を削除」，「構文要素の種類を置換」という操作を行うことで新たなテストデータを作成することができる．しかし，既存の技術 [1] では識別子の変更は可能だが，ソースコードを実行した際の振る舞いを変えることを目的としておらず，対応できない．よって，これらの情報を記述可能な入力形式を提案する．

2.2.2 変更に必要な情報

CDI ツールに対するテストデータの特徴を利用し，テストデータ毎の差異を変更することで新たなテストデータを作成するためには次の情報が必要である．

- 変更元となるテストデータ
  変更対象となる構文要素を含むテストデータの 1 つ
- 変更箇所
  変更元となるテストデータ中の差異となる構文要素を指定するための情報
- 変更内容
  変更箇所に対してどのような変更を行うかの情報

変更箇所と変更内容は，我々が提案する表を用いて入力する．

3 入力に表を用いたテストデータ生成支援ツール

本節では，テストデータ生成支援ツールの概要を示し，表を用いた入力に必要な情報をさらに整理する．本手法の特徴として，変更に必要な情報の入力に表を用いることが挙げられる．

3.1 ツールの概要

CDI ツールのテストデータ生成支援ツールに対する要求として次のようなものがある．

- 入力されたテストデータに変更を加え，新たなテストデータを出力する
- テストデータに対する変更情報の入力を支援する
- 少ない入力で多くのテストデータを生成する

これらの要求からツールの概略は図 1 のようになる．

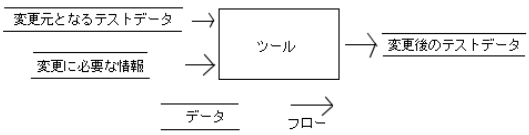

図 1　ツール全体の図

変更元となるテストデータは，「CDI ツールの検査対象となる構文要素を含む」，「構文規則に違反していない」，「Java のソースコード」であるということが条件であり，出力されるテストデータもまた「Java のソースコード」であり，「構文規則に違反していない」もののみが出力される．構文規則に違反するような変更が行われた場合，そのテストデータは出力されない．「CDI ツールの検査対象となる構文要素を含む」という条件は，CDI ツールの検査対象がなくては，検査対象に変更を加えて新たなテストデータを生成できないからである．変更に必要な情報は，変更を行う箇所とどのような変更を行うかという情報である．これらの情報を容易に入力可能な入力手法として，我々は表を用いた入力手法を提案する．

3.2 変更対象となる構文要素

CDI ツールの検査内容を分析し，変更する対象となる構文要素と変更に用いる操作を調査し，表での入力に必要な情報を整理した．

ソースコード中の構文要素を対象として変更を行うので，この対象となる構文要素を指定できる情報を入力する必要がある．対象となる構文要素の例としては，制御文 (ループ文・分岐文)，中置演算子，識別子等がある．

3.3 変更箇所の指定方法

変更箇所を指定する方法として，入力する表が次のものを表現できるよう設計する必要がある．

1. 構文要素
2. 構文要素間の関係

上記の 1,2 を組み合わせることで，さらに複雑な箇所の構文要素を指定することも可能である．我々は，AST を用いてテストデータを変更するので，これらの箇所を AST 上のノードやノード間の関係を用いて次のように記述できるようにする．

1. 構文要素
   AST 上のノードの型やノードの持つ値で表現
2. 構文要素間の関係
   AST 上のノードの親子関係，子孫関係で表現

よって，我々は変更箇所を AST 上のノードの型，値，親子関係，子孫関係を用いて指定できるようにする．

3.4 テストデータに加える操作

構文要素を変更する際，構文規則に違反しない変更が必要である．我々は，抽象構文木 (Abstract Syntax Tree 以下 AST) を利用する方法を提案する．AST とはプログラム言語等の構文規則中で本質的または抽象的な構文構造に基づく抽象構文を木構造で表現したものであり，各ノードごとに決められた親子関係を維持することで，構文規則に違反しない変更を可能にする．特に，Eclipse[3] が使用している AST を用いる．一般的に木を編集する操作としては，「代入 (ノードのラベルを別のラベルへ置き換え)」「挿入 (ノードの挿入)」「削除 (ノードの削除)」が文献 [4] では挙げられている．これらの操作を複合することで「置換 (削除＋挿入)」「部分木の削除 (削除の繰り返し)」「部分木の追加 (挿入の繰り返し)」が可能である．よって，2.2.1 節で述べたように，テストデータに対して，「構文要素を削除」，「構文要素の種類を置換」という操作が必要となるので，我々は次を AST に対して行う操作として記述可能にする．

操作の定義

- 削除
  対象となるノード一つを削除
  – ノードの削除

対象となるノード以下を削除
  – 部分木の削除
- 置換
  ノードの値を置換
  – 代入 (値の変更)
  ノードの種類を置換
  – 置換 (挿入＋削除)

4 表を用いた入力手法

4.1 表の定義

我々が提案する表の定義と記述方法について述べる．
3.2 節，3.3 節，3.4 節で変更に必要な情報を整理した結果，次の情報をツールに入力する．

- 対象となる構文要素
- AST 上の親子関係，子孫関係
- 操作の種類
- 変更後のノードの型や値

これらの情報を図 2 で示す形式に従った表を用いて入力する．

親ノード	対象	操作	変更後のノード

図 2　表の形式を表した図

変更箇所

- 親ノード：操作対象の親ノード
  – 操作対象に制限を加える場合に操作対象の親ノードを記述する．
- 対象：操作を行う対象
  – 特定の位置の構文要素を指定する場合は，Statement か Expression を記述し，ノードの値を操作する場合は値を保持するノードを記述する．

変更内容

- 操作：対象に行う操作
- 変更後のノード：置換後のノードの型，値
  – 部分木の削除, ノードの削除の場合は記述しない

4.2 表の記述

4.2.1 変更箇所の記述方法

本ツールでは，ノードの親子関係やノードの型を記述することで変更対象を指定する．変更対象となるノードの親子関係を指定しない場合は，対象ノードの親ノードを記述する必要は無い．

4.2.2 変更対象に制限を加える場合の記述方法

変更対象とするノードに制限を加える場合は，表の 1 行目に親子関係のみを記述する．さらに制限を加えていく場合は，親子関係を行毎に記述していき，最後の行に対象と変更内容を記述する．行と行との関係は，親子関係と親子関係との間に 0 個以上のノードが存在する事を意味する．

4.2.3 変更内容の記述方法

変更内容には，3.4 節で定義した次の操作の種類を記述する．

- 削除
  部分木の削除
  対象ノード以下を全て削除
  ノードの削除
  対象ノードのみを削除
- 置換
  ノードの置換
  対象ノードを指定したノードの型に置換
  ノードの値の置換
  対象ノードの値を指定したノードの値に置換

置換の操作を行う場合には置換後のノードの型やノードの値を記述する．数値等に変更する場合は任意の個数への置換が必要になるので，1 つの表につき任意の数記述することが出来る．
複数のノードの型を一括で記述する場合は 1 つのセルに 1 種類のまとめたノードの型を記述する．また，While 文，For 文，Do 文の繰り返し文を LoopStatement，If 文，Switch 文の分岐文を BranchStatement と定義し，対象や変更後のノードとして記述できるようにする．

4.2.4 表の記述例

例として挙げる変更元となるテストデータに対して変更する場合の記述例を示す．

変更元となるテストデータの例

```
1: public static void main(){
2:   int a = 0;
3:   while(a < 3){
4:     a++;
5:   }
6:}
```

While 文のループ回数を検査する機能に対するテストデータを作成する場合,「While 文の条件式」の「中置演算子の右辺」の「数値リテラル」を変更することで，テストデータを作成できる．このような変更を行う場合，表 1 のように記述する．

表 1　While 文のループ回数を検査する CDI ツールの機能に対するテストデータの記述例

親ノード	対象	操作	変更後のノード
WhileStatement	Expression		
InfixExpression	Expression(Right)		
	NumberLiteral	代入	4
			5

1 行目では While 文の条件式を指定している．この条件

式に対し，2 行目で，条件式中の中置演算子の右辺を指定する．3 行目で，これらの条件を満たす中置演算子の右辺が数値リテラルの時，その値を 4 と 5 に置換する．変更元のテストデータ中でこの条件を満たす数値リテラルは 3 行目の 3 なので，この数値が変更される．
ループ文を検査する CDI ツールの機能に対するテストデータを作成する場合，「While 文」を「他のループ文」に変更することで，テストデータを作成できる．このような変更を行う場合，表 2 のように記述する．

表 2　ループ文を検査する CDI ツールの機能に対するテストデータの記述例

親ノード	対象	操作	変更後のノード
	WhileStatment	置換	LoopStatement

1 行目で，全ての WhileStatement を指定している．変更元のテストデータ中でこの条件を満たす数値リテラルは 3 行目の While なので，この While が変更される．LoopStatement は，For 文，Do 文，While 文をグループ化した単位であり，この 3 種の文に置換を行う．

4.3　表の入力

複数箇所を変更する場合は，1 箇所の変更につき 1 つの表を入力する．複数の表を入力した場合，全ての変更内容を入力順に組み合わせた複数のテストデータを出力する．

5　考察

5.1　他の入力手法との比較

本手法では変更元となるテストデータとテストデータの仕様を入力し，新たなテストデータを生成する．他の入力手法として，テストデータの仕様のみを与えるという方法も考えられた．しかし，この方法を用いてテストデータを生成する場合，変更箇所以外の情報が必要になり，ソースコードと同等の情報を記述することになってしまい，記述量が増えるので非効率的と言える．

5.2　CDI ツールのテストデータ生成に必要な情報を表現可能かの検証

本手法は AST のノードの親子関係，子孫関係を用いることで，CDI ツールの検査箇所となる一部の構文要素を指定できる．CDI ツールのソースコードの構造に対する検査の機能をテストするテストデータを生成する場合，一部の差異となる構文要素を変更することで，新たなテストデータを生成することが可能である．また，制御フローやデータフローに対する検査の機能をテストするテストデータも，テストデータ毎の差異は一部の構文要素のみである．本ツールのユーザが制御フロー，データフローを考慮する必要があるが，構文要素を指定して変更を行うことで新たなテストデータを生成可能である．よって，本手法を用いる事で CDI ツールのテストデータ生成に必要な変更箇所を指定できるといえる．

5.3　metal との比較

我々が提案した表と，metal の記法を用いて 4.3 節の表 1 を記述した場合と記述量を比較する．

表 1 を metal で記述した場合

```
start: { WhileStatement } ==> InWhile;
InWhile: { Expression } ==> WhileExpr;
        | { Statement } ==> Stop;
WhileExpr :{ InfixExpression } ==> ExprInfx;
ExprInfx :{ Expression(Right) } ==> InfxExpr_r;
InfxExpr_r :{ NumberLiteral } ==> Stop,
{ 置換 (4,5);}
;
```

metal では状態を記述するのに対し，本手法では状態を記述しない．表に記述したノードの名称自体が状態を意味するので，本手法のように親子関係，子孫関係を記述する場合においては明示的に状態を記述する必要はない．

6　おわりに

我々は入力されたソースコードを変更し，変更されたソースコードを出力するテストデータ生成支援ツールを提案した．本ツールに対する入力方法について研究を行い，CDI ツールに対するテストデータを分析することで，変更に必要な情報を整理し，ツールのユーザが容易に入力できる記法として表を用いた入力手法を提案した．これにより言語を習得する手間がなくとも，変更箇所を指定できる記法を提案した．
今後の課題として，整理した情報を元に入力支援を行う GUI の設計があげられる．

参考文献

[1] 丸山勝久, “ XML を用いた拡張性の高いリファクタリングツール, ” 電子情報通信学会論文誌, vol. J88-D-I, no. 2, 2005, pp. 175-185.

[2] B. Chelf, S. Hallemand, Y. Xie, and D. Engler, “ A System and Language for Building System-Specific, Static Analyses, ” *In Proceeding of the ACM SIGPLAN 2002 Conference on Programming Language Design and Implementation*, 2002, pp. 69-82.

[3] Eclipse JDT API Specification, *Eclipse.org*, 2009; www.eclipse.org/.

[4] 久保山哲二, 宮原哲治, “ 木の編集距離を用いた半構造データからの情報抽出, ” 第 18 回人工知能学会全国大会講演論文集, 2004, pp. 4.