4L-9

形式的仕様記述に基づくテストデータの生成

亀田　佑二† 　小倉 崇† 　林 雄二‡
北海道情報大学経営情報学研究科† 　北海道情報大学経営情報学部システム情報学科‡

１．はじめに

形式的技法に基づき、プログラムの自動生成、段階的詳細化、正当性の証明などが可能になれば、誤りのないプログラム開発が実現されることが期待される。しかし、現状では、実際的なプログラムに対しそれらを実現することは困難である。

現状で実現可能な方法として、作成されたメソッドの正当性をテストすることに形式的仕様を活かすことがあげられる。そのための手段として、仕様そのものからテストデータを自動生成し、人手で作成されたメソッドが仕様を満たしているかどうかの検証を、テストによって行うことが考えられる。

我々は、形式的仕様記述[1]から出発して、ブラックボックステストを通してエラーを検出しながらプログラム開発を進めていくためのツール開発を進めている。以下の図１は、形式的仕様に基づくメソッドの正当性を検証するために、我々が開発を進めているシステム[2]の構成図である。

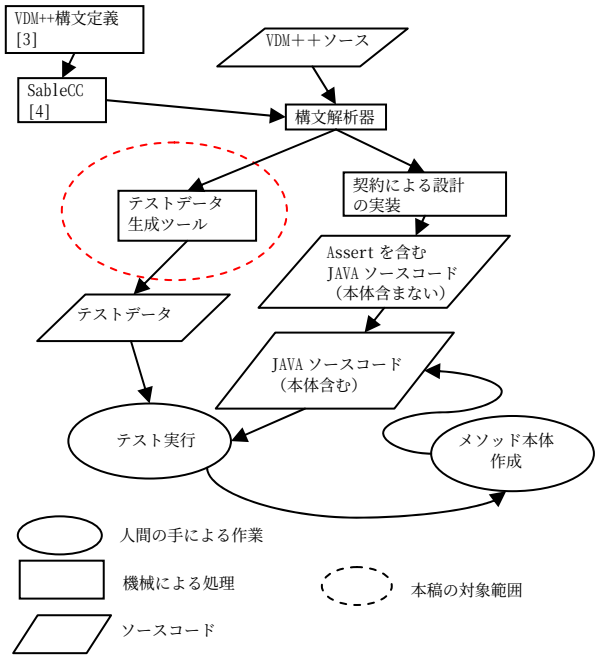

図１．形式的仕様を活かすシステム

まず、形式的仕様（VDM++ソース）に対して、構文解析を行い、この解析結果を元にテストデータの生成と、契約による設計に基づく JAVA ソースコードのスケルトンの生成を行う。さらに、このスケルトンにメソッド本体を付け加えることで、JAVA ソースコードを完成させる。

このソースコードに対して、生成されていたテストデータによる Unit テストを行う。これにより形式的仕様に基づいたメソッドが作成されているかどうかの検証が可能となる。

関連研究として、文献[5]では、VDM-SL からのテストケースの作成理論を提案しているが、これはテストケースを生み出すことはできるが、実際のテストデータを生成するものではない。

本稿では、このシステムの一部分であるテストデータ生成ツールに関して報告する。

２．VDM++からテストデータの生成

VDM++[3]によって記述された各プログラムの仕様からテストデータを作成し、そのテストデータに従って、テスト駆動開発の形でメソッドを作成していくことで、仕様から外れることのないプログラムの作成が可能となる。

テスト駆動開発の中にテストファースト[6]という考え方がある。テストファーストとは、プログラムより先にまず、テストデータをコーディングし、それからそのテストデータに従ってプログラムを作成していくという手法である。この方法で作成していく場合は、テストデータをプログラムの仕様よりも先に作成していくということで、工数の増大が起こる可能性が高い。この問題を解決するために、プログラムの仕様そのものからテストデータの自動生成を行うことで、各メソッドの正当性を保証することが可能となる。

これにより、プログラムの仕様の作成＝テストデータの作成となり開発工数の削減にも役立つと考えられる。

具体的に VDM++からのテストデータ生成を行なうために、まず事前条件、事後条件の式を選言標準形に変換する。そして、各項に対して境界値分析を行いテストデータの生成を行う。

３．テストデータ生成の実現方法と具体例

事前条件はメソッド呼び出し側が満たすべき条件ではあるが、メソッドの入口で不正データを排

『Test Data Generation based on Fomal Specifications』
†Yuji Kameda　†Takashi Ogura
Guraduate School of Hokkaidou Information University
‡Yuji Hayashi
Systems and Informatics,Hokkaidou Information University

除するべきであると考え、事前条件を満たすデータと満たさないデータをテストデータとして生成することにした。事後条件については、未知の値がその条件内に含まれている。そこで、未知の値を含まない（即ち、仮引数とインスタンス変数のみの）項に対して、境界値分析でテストデータを生成することにした。

３．１．数値型のテストデータ

pre 式がすべて and の場合はすべての組み合わせを考慮する。しかし、or を含む場合は、選言標準形に直して or で区切って各項ごとにテストデータを生成する。

例１）pre 式に対するテストデータの生成

```
functions
  example1(a:int,b:int)c:int
    pre a>10 and b>0  or a<30 and b<5
```

この場合、a,b のテストデータの候補値は上記と同様の考えかたでテストデータに追加する。ただし、今回のケースの場合は、pre 式が、a>10 and b>0 or a<30 and b<5 となっている。つまり条件は、(a>10 かつ b>0）または(a<30 かつ b<5)となる。したがって、生成すべきテストデータは、

(a>10 かつ b>0）より、

a × b ⊃ {(10,0),(10,1),(11,0),(11,1)}

(a<30 かつ b<5) より、

a × b ⊃ {(30,5),(30,4),(29,5),(29,4)}

が生成される。したがってすべてを含めると、

a × b ⊃ {(10,0),(10,1),(11,0),(11,1),(30,5),(30,4),(29,5),(29,4)}

の８通りのテストデータが生成される。

例２）post 式に対するテストデータの生成

```
functions
  example2(a:int,b:int)c:int
    pre  a>0 and b>0
    post a>b and c=a*a - b*b  or
         a<=b and c=b*b-a*a
```

pre のテストデータ

a × b ⊃ { (1,1), (1,0), (0,1), (0,0)}

post のテストデータを生成するために条件式を選言標準形に変換すると以下のようになる。

a>0 and b>0 and a>b and c=a*a - b*b or
a>0 and b>0 and a<=b and c=b*b-a*a

pre 条件を満たす範囲で、境界値分析でテストデータを生成する

post のテストデータ

a>0 and b>0 and a>b より

a × b ⊃ { (1,1), (2,1)}

a>0 and b>0 and a<=b より

a × b ⊃ { (1,1), (2,2), (2,1)}

重複したデータは、もちろん１つにまとめる。

post は pre を満たした条件のテストデータ以外は取り入れない。したがって、前述に述べたように、post の条件に pre の条件を含めたものをテストデータとして生成する。

３．２．文字列型のテストデータ

テストで正常に通る文字列があらかじめ仕様に記述されている場合、テストデータは、それを含めてほかに通したくない場合の文字列を一組ランダムに生成した文字列を組み合わせたものをテストデータとして生成する。

例３）ユーザーが記述する仕様記述

```
instance variable
  testExampleA = { “asa”,”hiru”,”bann” };
operations
  example3(a:String) c:String
    rd testExampleA
    pre a in set testExampleA
```

テストデータ＝{“asa”,”hiru”,”bann”,ランダムに生成した文字列} となる。

仕様にあらかじめ記述されていない場合は、テスト実行時にユーザーに通したい文字列を入力してもらいそれに一組ランダムに生成した文字列を組み合わせたものをテストデータとして生成する。

したがって、テストデータ＝{ユーザーが入力した文字列,ランダムに生成した文字列}となる。

４．おわりに

テストデータに対する結果の正当性チェックには、事前，事後条件に対する自動判定が必要である。我々は、VDM++ソースから、事前，事後，不変条件の自動判定が可能な Java ソースコードを生成する機能の開発も進めている。本稿のテストデータ生成は、そのような機能を背景として活かされるものである。

現段階では、基本データ型に限定して開発を進めているが、VDM++には多様なデータ型があり、Java のクラスに対応させてどのようにテストデータとして与えるか、今後の課題として残されている。

参考文献

[1]荒木啓二郎，張漢明 : プログラム仕様記述論，オーム社(2002)
[2] 小倉崇，亀田佑二，林雄二：VDM++から Java の契約による設計へ,情報処理北海道シンポジウム （2006）
[3]vdm++サイト:http://www.vdmtools.jp/ （2006）
[4]SableCC サイト:http://sablecc.org/ (2006)
[5]Meudec,C：Automatic Generation of Software Test Cases From Formal Specifications, phd thesis, The Queen's University of Belfast(1998)
[6]今野睦：JUnit によるテストファースト開発入門,サイバービーンズ株式会社（2004）