Javaソースコードの CDI(Code Inspection) ツールの開発
～コードインスペクションツールに対するテストプロセスの改善～

M2008MM024 二宮剛史
指導教員：野呂昌満

1　はじめに

ソフトウェアを実行しないで検証する静的検証の 1 つとしてインスペクションがある．プログラムコードに対するインスペクションをコードインスペクションと呼ぶ．

我々の OJL(On the Job Learning) では，これまでの OJL で開発されたコードインスペクションツール [3, 4](以下，CDI ツール) を引き継ぎ，開発した．CDI ツールの開発は，プロダクトラインソフトウェアエンジニアリング [2](以下，PLSE) に基づきおこなわれた．開発を通して生み出された核資産は，アーキテクチャ，抽象構文木，フローグラフ，検査項目の開発プロセスなどである．我々の開発では CDI ツールに対して機能の改善および品質の向上をおこなった．

CDI ツールの各検査項目に対するテストでは，入力となるテストデータはソースコードである．ソースコードが抽象構文木や制御フロー，データフローに変換され，そのツリー表現やグラフ表現をもとに検査がおこなわれる．テスト担当者は，テスト対象の検査の仕様をもとに様々なツリー表現やグラフ表現となるソースコードを多数作成する必要がある．

CDI ツールの開発は PLSE に基づきおこなわれているので，開発時に作成されたテストデータやテストデータの作成からテストの実施までのテストプロセスも核資産である．検査項目は，抽象構文木や制御フロー，データフローを利用して検査をおこなうので，テストデータとなるソースコードは抽象構文木の構文要素や，制御フロー，データフローについて考慮し，これらを組み合わせて作成する必要がある．組み合わせの数が多数存在することから，テストデータ作成に労力が掛かる．本研究では，核資産の品質向上を目的として，テストプロセスの改善をおこなった．

テストプロセスを改善することで，テストデータ作成に掛かる労力の削減をおこない，テストデータの生産性を向上させる．テストデータ作成の労力削減やテストデータの生産性を向上させる目的で，テストデータ作成方法を定義した．今まで作成してきたテストデータからテストデータとなるソースコードの固定部分と変動部分を整理し，変動部分の変更方法をテストデータ作成方法のカタログとして一覧化をおこなった．テストデータ作成カタログとは，検査項目の仕様をテストデータ用にカスタマイズするために，テストデータとなるソースコードを作成するために必要とする着目対象，分類，詳細な内容を一覧化したものである．テストデータ作成方法のカタログを作成し，テストデータ作成カタログを用いたテストプロセスを構築することでテストプロセスの改善をおこなった．

本研究のメタ技術は CDI ツールの開発および，テストデータ作成カタログの作成に用いた PLSE である．

ベース技術は，ソフトウェアテスト方法 [1]，プログラム解析技術である．

専用手法は CDI ツールに対するテストプロセスの改善である．

2　関連技術

2.1　PLSE

PLSE とは系統的に再利用をおこない，ソフトウェアを開発する方法論である．

PLSE は 3 つの活動によって構成される．3 つの活動を図 1 に示す．

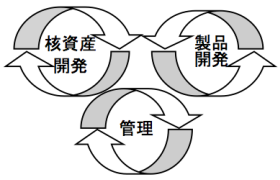

図 1　PLSE の 3 つの活動

PLSE では核資産をもとに部品を組み合わせて製品開発をおこなう．

核資産開発は，開発の成果である要求分析の結果やアーキテクチャなど，製品系列で再利用可能な部品を構築する活動である．製品開発は，核資産を利用して実際の製品を開発する活動である．管理は，核資産開発や製品開発の活動をどのようにおこなっていくかを決定する活動である．

3 つの活動がお互いに関連しながらソフトウェアの開発がおこなわれる．

2.2　ソフトウェアテスト方法

ソフトウェアテストはソフトウェアに求められる要求に対する動作や機能の保証が目的でおこなわれる．テスト対象やテストをおこなう目的により，実施するテストの種類 (単体テストやシステムテストなど)，テスト方法 (ホワイトボックステスト，ブラックボックステストなど) を選択する．テスト設計において，ソフトウェアの特徴を把握し，適切なテスト方法を選択することでソフトウェアの品質保証をおこなう．

テストは要求に対する保証と欠陥の検出が目的である．ソフトウェアに欠陥が含まれていないことを保証することは出来ないが，欠陥を検出することで欠陥が存在することを示すことが出来る．ソフトウェアテストでは，テストケースの数を削減しながら効率的に欠陥を検出することが求められる．

3　CDI ツールに対するテスト

3.1　現在おこなわれているテスト方法

現在のテストプロセスでは，CDI ツールの検査項目に対するテスト方法はブラックボックステストが実施されている．図 2 にテストデータとなるソースコード作成において，テストデータ作成者が考慮している要素を示す．

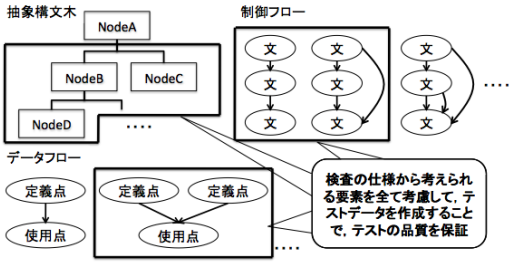

図 2　ブラックボックステストにおけるテストデータ作成

テスト対象である検査項目は，抽象構文木や制御フロー，データフローを利用して検査をおこなう．テストでは，CDI ツールにテストデータとなるソースコードを入力し，CDI ツールから出力される検査結果が期待する結果と同じ結果となるかどうかを確認する．期待する結果とは，欠陥の可能性がある箇所が検出される，または検出されない結果のことである．

検査項目はソースコードを表現する抽象構文木や制御フロー，データフローの情報を組み合わせて，欠陥の可能性がある箇所を発見することから，検査項目のテストに用いるソースコードは，検査項目の仕様から考慮される構文要素全てや制御フロー，データフローを組み合わせて作成される．

3.2　テストデータ作成における問題点

検査項目に対するテストのテストデータ作成における問題は，構文要素や制御フロー，データフローを組み合わせることで，多数のテストデータを一から作成する点である．CDI ツールの検査項目に対するテストデータはソースコードである．ソースコード上には構文要素や制御構造，データの流れが表現されている．テスト担当者はテストデータを作成する際に，ソースコード上で制御フローを考え，その制御フローにおいて変数の定義や使用といったデータフローを考える必要がある．さらに，制御フロー上に出てくる構文要素についても考える必要がある．これらの情報が組み合わさってテストデータとなるソースコード群を作成するので，テスト担当者の労力が大きい．作成したテストデータはテスト対象の検査に特化したものになっており，他の検査のテストデータとして再利用することは難しく，検査ごとにテストデータを作成する必要がある点も労力が大きい原因である．

オブジェクト変数に null 値が格納されている場合にオブジェクト変数を参照していないか調べる検査についてのテストデータ例を図 3，図 4 に示す．

図 3 では，オブジェクト変数 s について，変数 s の値に

```
public void testMethod1() {
    String s = null ;

    s = "sample" ; // sの定義点
    // s.equals()は警告対象外
    if ( s.equals("sample") ){
        ・・・
    }
}
```

図 3　テストデータ例 1

```
public void testMethod2() {
    String s = null ; // sの定義点1
    if ( ・・・ ) {
        s = "sample" ; // sの定義点2
    }
    // s.equals()は警告対象
    if ( s.equals("sample") ){
        ・・・
    }
}
```

図 4　テストデータ例 2

null 以外の値を代入した後に，変数 s を使用した場合のテストデータである．

図 4 では，テストデータ例 1 の場合に制御フローを追加し，オブジェクト変数 s のデータフロー上で null 値が格納されているときに変数 s を使用するという関係を表現したテストデータである．

4　テストデータ作成カタログ

テストデータ作成カタログとは，検査項目の仕様をテストデータ用にカスタマイズするために，テストデータとなるソースコードを作成するために必要とする着目対象，分類，詳細な内容を一覧化したものである．

テストデータ作成には，今までの開発の中で作成してきたテストデータからテストデータをどのような基準で作成したら良いかまとめたテストデータ作成カタログを作成することで，労力の削減が見込める．

我々は今までの開発を通して幾つものテストデータを作成してきた．テスト対象の検査項目の仕様から直接テストデータを自動生成することは難しいが，今までのテストデータ作成経験を整理，一覧化することで，新たなテストデータ作成時に役立てることが出来る．

今までに作成されたテストデータ群をテストデータとして変動していく部分に着目して整理することで，テストデータ作成カタログに記載する情報を一覧化する．テストデータ作成カタログには着目対象，分類，詳細を記載する．着目対象には追加，変更する対象を記述する．分類には，制御フロー，データフロー，型検査のどれに属するのかを記述する．詳細には，拡張方法の実現例を記述する．テストデータ作成カタログの例を図 5 に示す．

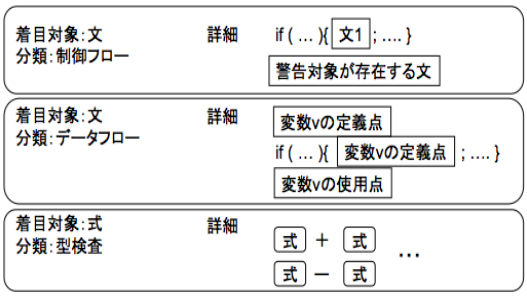

図 5　テストデータ作成カタログの例

テストデータは構文規則や制御フロー，データフローを考慮して作成するので，テストデータ作成カタログにはこの 3 つのうちのどれについての作成方法なのかという情報が必要になる．新しいテストデータとなるソース

コードは，既にテストデータとして作成したソースコードに記述されている文や式が変化したものなので，文に着目した作成方法なのか，式に着目した作成方法なのかという情報も必要である．

カタログを用いたテストデータの自動生成

カタログ化したテストデータ作成方法のうち，テストデータの自動生成が可能なテストデータ作成方法と不可能なテストデータ作成方法を分類する．

テストデータとなるソースコードの自動生成は，生成するソースコードの雛形となるソースコード，雛形のソースコードの変更箇所，変更後のソースコード片が揃っていれば可能である．雛形となるソースコード内の記述のうち，変更する記述を指定し，別の記述に置き換えることで新たなソースコードを作成することができる．テストデータ作成カタログにはテストデータの拡張方法が記載されている．テストデータ作成カタログに記載されている情報を元にすることで，テストデータとなるソースコードの自動生成が可能なテストデータ作成方法がある．図 6 に自動生成可能なテストデータ作成方法と不可能なテストデータ作成方法の例を示す．

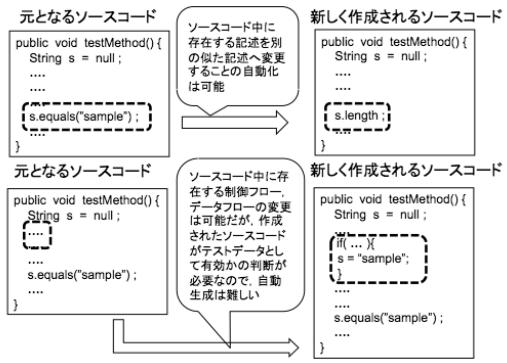

図 6 自動生成が可能なテストデータ作成方法と不可能なテストデータ作成方法の例

自動生成可能なテストデータ作成方法は，拡張するソースコードに既に記述されている構文要素を別の構文要素に変更していく作成方法である．例えば，オブジェクト変数の参照として，オブジェクト変数に格納される値の参照やオブジェクト変数に対するフィールドアクセス，メソッド呼び出しがある．新しいテストデータは，テストデータとなるソースコードの雛形を元にして作成される．この雛形となるソースコードにオブジェクト変数の参照の記述があれば，別のオブジェクト変数の参照方法へと変換することが出来る．

自動生成不可能なテストデータ作成方法は，制御フローやデータフローに関する作成方法である．開発対象の CDI ツールでは，制御フローやデータフローはフローのノードが文単位で扱われているので，文を変更箇所として指定し，他の制御フローやデータフローに置き換えることで入力となったソースコードの制御フローやデータフローとは異なる制御フロー，データフローが作成できる．しかし，作成できた制御フローやデータフローがテスト対象である検査項目にとって，有効なテストデータとなるかは，ソースコードの意味だけでなく，検査項目の仕様まで考慮しなければいけない．よって，制御フローやデータフローに関するテストデータ作成方法は，テスト対象に取って意味のあるテストデータの自動生成が不可能であると考える．

5 テストプロセスの再構築

テストデータ作成カタログを用いたテストプロセスを現在のテストプロセスをもとに構築した．

新しいテストプロセスを図 7 に示す．

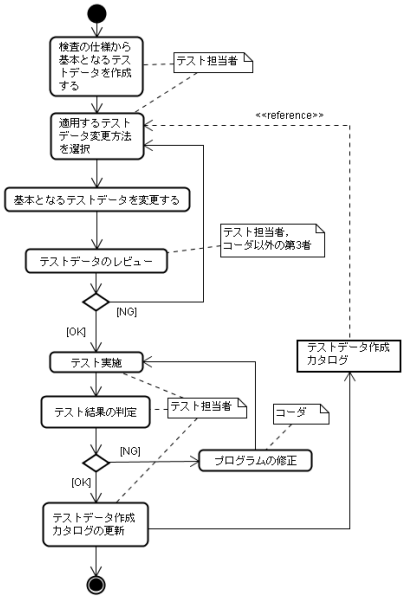

図 7 新しいテストプロセス

テストデータの作成は，テスト対象の CDI ツールの検査項目の仕様からテストデータの雛形を作成し，その雛形をテストデータ作成カタログを用いて変更していくことで様々なテストデータを作成する．改善前のテストプロセスとの違いは，テストデータ作成カタログを参照しながら，テストデータを作成する部分とテストデータ作成カタログを更新する部分が追加されたことである．テストデータ作成においてカタログを参照しながらテストデータを作成することで，テスト担当者個人以外が考えたテストデータの変更方法をテストデータ作成に応用出来る．

6 考察

6.1 構築したテストプロセスの有用性の考察

構築したテストデータ作成カタログを用いたテストプロセスの妥当性を確認するために，検査項目の仕様が変更された場合に，テストデータ作成カタログを用いたテストプロセスの有用性について考察した．我々の OJL では CDI ツールの機能の改善をおこなっているので，新しい検査項目の追加だけでなく，既存の検査項目の修正もおこなわれる．既存の検査項目の修正では，既存の検査項目に対して追加される仕様が存在する．

テストデータ作成カタログとして，テストデータの作成方法を整理してあるので，検査項目の仕様が変更された

検査項目に対してのテストデータ作成がおこないやすい．図 8 に仕様が変更された検査項目に対するテストデータ作成の手順を示す．

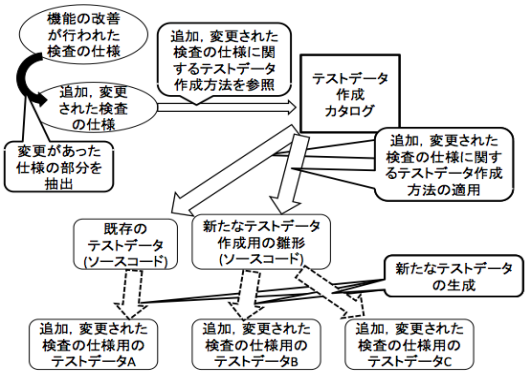

図 8 仕様が変更された検査項目のテストデータ作成

仕様が変更された検査項目から，追加や変更された検査の仕様が制御フロー，データフロー，型検査のどれに関するものなのか，文または式に関するものなのか判断することで必要とするテストデータの作成方針が決まる．作成方針が決まった後は，テストデータ作成カタログに記載されているテストデータの拡張方法で，既存のテストデータであるソースコードや新たに作成したテストデータの雛形として作成したソースコードを拡張していくだけで，テストに必要なテストデータが作成できる．テスト対象である検査の仕様に変更があった場合にも，テストデータの作成がおこないやすいので，テストデータ作成カタログを用いたテストプロセスは有用であると言える．

6.2 他のドメインへのテストデータ作成カタログを用いたテストプロセスの適用可能性の考察

構築したテストプロセスの他のドメインへの適用可能性を考察した．

CDI ツールの入力となるデータの特徴として，ツールへの入力データがプログラミング言語で記述されたソースコードである点があげられる．同じ意味を持つソースコードの表現方法が無数に存在することから，テストデータとなるソースコードの作成方法を一覧化したカタログを用いたテストプロセスが CDI ツールには有効である．

他の言語処理系として，コンパイラを例として考える．コンパイラは，ある規則に従って記述された文字列 (ある言語で記述されたソースコード) を入力として，字句解析，構文解析処理をおこなう．構文解析処理をおこなった後に中間系を生成し，中間系をもとにして別の規則に従って記述された文字列 (入力されたソースコードの記述に使用された言語とは異なる言語で記述されたソースコード) を出力する．

コンパイラに対するテストでは，字句解析器や構文解析器といったコンパイラを構成するモジュールに対するテストがある．字句解析器や構文解析器といったモジュールの動作をテストするには，実際にソースコードを入力して確認する必要がある．入力となるソースコードは，字句解析器や構文解析器で扱うべき要素を網羅して作成する必要がある．字句解析器や構文解析器モジュールに対するテストデータ作成カタログを用いたテストプロセスを図 9 に示す．

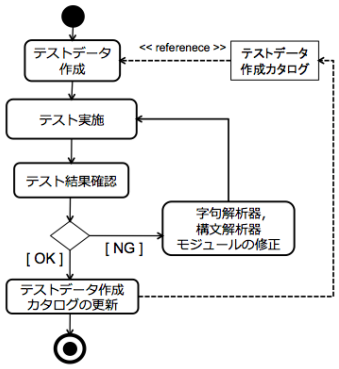

図 9 字句解析器や構文解析器モジュールに対するテストデータ作成テストプロセス

入力となるテストデータの作成に，テストデータとしてどのようなソースコードを作成したらよいかという指針として，テストデータ作成カタログを用いることで，網羅的に作成する必要があるテストデータの作成が効率的におこなえる．

よって，他の言語処理系へもテストデータ作成カタログを用いたテストプロセスの適用可能性がある．

7 おわりに

本研究は，CDI ツールの開発に対するテストプロセスの改善をおこなった．テストプロセスの改善では，テストデータ作成カタログを用いたテストプロセスを構築した．構築したテストプロセスに対しての妥当性の考察と他のドメインへのテストデータ作成カタログを用いたテストプロセスの適用可能性について考察をおこなった．

本研究のメタ技術は CDI ツールの開発および，テストデータ作成カタログの作成に用いた PLSE である．

ベース技術はテストデータ作成のために必要とするプログラム解析技術，テストプロセス構築のために必要なソフトウェアテスト方法である．

参考文献

[1] B. Boris, *Software Testing Techniques, Second Edition*, Van Nortrand Reinhold, 1990.

[2] L. M. Northrop, “ SEI ’s Software Product Line Tenets, ” *IEEE Software*, vol.19, no.4, pp. 32-40, 2002.

[3] 後藤洋, “ Java ソースコードの CDI(Code Inspection) の開発 ～アーキテクチャの構築～, ” 南山大学大学院 数理情報研究科 2008 年度 修士論文要旨集，pp.130-133，March 2009.

[4] 立岑将輝, “ Java ソースコードの CDI(Code Inspection) の開発 ～フローグラフとその処理の設計実現～, ” 南山大学大学院 数理情報研究科 2008 年度 修士論文要旨集，pp.144-147，March 2009.