2B-04

テスト自動化技術を利用したマトリクスチェックリストに基づくテスト方法

†鹿糠 秀行　†大林 浩気　‡内海 幸也　‡武藤 邦弘

†株式会社 日立製作所 研究開発グループ　‡株式会社 日立製作所 アプリケーションサービス事業部

1 はじめに

テストケースの抜け漏れを防ぎテストの網羅性を高めるために，テストケースをマトリクスチェックリスト形式で設計し，これに基づいてテストする方法がある．

本論文では，記号実行を用いたテスト入力値生成技術 [1] でテスト対象のソースコードからマトリクスチェックリストの一部自動生成を可能にし，さらにマトリクスチェックリストから実行可能なテストコードを自動生成することで，マトリクスチェックリストに基づいたソフトウェアの単体テストを効率的に行う方法を提案する．

2 マトリクスチェックリストとテスト方法

2.1 マトリクスチェックリスト

本研究で規定するマトリクスチェックリストとは，以下の **(a)** から **(d)** で構成されるテスト対象のプログラムを単体テストするためのテスト設計書である．図 1 にマトリクスチェックリストのイメージを示す．

(a) テスト対象のメソッド情報：テスト対象のパッケージ名，クラス名，メソッド名からなる．

(b) テストの入力値：テスト対象のメソッドに与えるデータ項目とその値のことである．

(c) テストの期待値：テスト対象のメソッドに (b) で定めたテストの入力値を与えてテスト実行した結果，期待する戻りのデータ項目とその値，または発生が期待される例外のクラスとメッセージのことである．

(d) スタブ設定：テスト対象のメソッドが他のクラスのメソッドを参照する場合がある．単体テストにおいて参照するクラスとしてスタブを用いる場合に定義する．

(b) と **(c)** の 1 組が 1 つのテストケースを表現し，テスト入力値の組み合わせによってテストケースが増えることでマトリクスが形成される．なおこのマトリクスは一般にディシジョンテーブルと呼ばれている．

2.2 マトリクスチェックリストに基づく単体テスト方法

マトリクスチェックリストに基づいて単体テストする場合の流れを以下に記す．

パッケージ名	sample
メソッド名	executeSampleMethod
スコープ	public
戻り値	データ型
	dto.SampleDto

				テストケース		
入力値	項目名	データ型		001	002	…
	item1	int		1	…	…
	item2	java.util.BigDecimal		111	…	…
	:	:		:	:	:
期待値(戻り値)	項目名	データ型				
	itemA	int		2	…	…
	itemB	java.util.BigDecimal		222	…	…
	:	:		:	:	:
期待値(例外)	例外クラス			…	…	…
	メッセージ			…	…	…
スタブ設定	パッケージ名	クラス名	メソッド名			
	sample	class1	execute	【Stub】c1	…	…
	sample	class2	execute	【Stub】c2	…	…
	:	:	:	:	:	:

図 1: マトリクスチェックリストのサンプルイメージ

(1) マトリクスチェックリストの作成：テスト対象のプログラムの元になった設計書類に基づいて，テスト対象のメソッド情報を記入し，テストの入力値とそれに対する期待値を定めることでテストケースを定義し，必要に応じてスタブ設定を定義し，マトリクスチェックリストを作成する．

(2) テストコードの作成：マトリクスチェックリストに基づいてテストコードを作成する．

(3) テストの実行：テストコードを実行して，テスト対象のプログラムがマトリクスチェックリストで定めた入力値に対して期待値を得られるかを確認する．

3 テスト自動化技術の適用

マトリクスチェックリストを作成する工数はテスト対象のプログラムの引数と戻り値のデータ項目数が多いほど増加し，マトリクスチェックリストに基づいてテストコードを作成する工数も増加する．

そこで，マトリクスチェックリストに基づく単体テストを効率化するために，テスト自動化技術としてテスト入力値生成技術を開発・適用しマトリクスチェックリスト中のテストの入力値を自動生成することでマトリクスチェックリストの作成工数を軽減し，マトリクスチェックリストから実行可能なテストコードの生成機能を開発・適用しテストコードの作成工数を削減する方法を提案する．

3.1 テスト入力値生成機能の開発と適用

テスト入力値を生成するツールとして，Symbolic Path Finder（SPF)[2] を採用した．SPF はテスト対象のプロ

A Testing Method Based on Matrix Check List by Using Test Automation Technology

†Hideyuki KANUKA　†Hiroki OHBAYASHI　‡Yukiya UTSUMI　‡Kunihiro MUTO

†Hitachi, Ltd., Research and Development Group

‡Hitachi, Ltd., Applications Services Division

グラムを入力に，記号実行技術によりプログラムの実行パスを網羅する入力値を出力することができる．

しかし通常の SPF には，オブジェクト型の変数を解析できない，アノテーションを解釈できない，データベースやファイルなど外部アクセスするプログラムを解析できない問題があった．そこで SPF で解析できる範囲を広げるために SPF へプログラムを入力する前に，オブジェクト型を同等のプリミティブ型へ変換し，アノテーションと同等のコードを生成，外部アクセス等を伴い解析できない API 呼び出しをスタブに置き換える前処理機能を開発した [3].

本機能を利用する前提として，テスト対象のプログラムがあること，またそれに対するマトリクスチェックリストが必要である．マトリクスチェックリストには，本機能がテスト対象を特定するためのメソッド情報とスタブ設定が記入されている必要がある．スタブ設定はテスト対象のプログラムの中で解析できない API 呼び出しをスタブコードに置き換えるために使用される．

本機能で生成されるテスト入力値は，マトリクスチェックリストのテスト入力値欄に自動記入される．ユーザは自動生成されたテスト入力値を確認し，入力値に対する期待値を設定する．なお，本機能で生成されるテスト入力値はプログラムの実行パスを網羅する入力値であり，ホワイトボックステストの観点での値であるため，必要に応じてテスト入力値とその期待値の追加が必要である．

3.2 テストコード生成機能の開発と適用

テストコードの生成機能では，テストコードに必要なテスト対象のメソッド情報，テストの入力値，テストの期待値，スタブ設定をマトリクスチェックリストから抽出して，単体テスティングフレームワークに準拠したテストコードを生成する．

本機能で生成されるテストコードは，スタブ設定の情報を元にスタブコードを生成し，テスト対象のプログラムの引数にテストの入力値を設定した上で呼び出し，実行後の結果を受け取りテストの期待値と比較することで単体テストの合否を自動判定できる．

3.3 テスト自動化技術を利用した単体テスト方法

以上で述べたテスト自動化技術を利用した単体テストの流れを図 2 に示し以下で説明する．

(1) マトリクスチェックリストの作成：テスト対象のプログラムの元になった設計書類に基づいて，テスト対象のメソッド情報を記入し (図 2-①)，スタブ設定を定義する (図 2-②).

テスト入力値生成機能によってマトリクスチェックリストにテスト入力値が生成される (図 2-③).

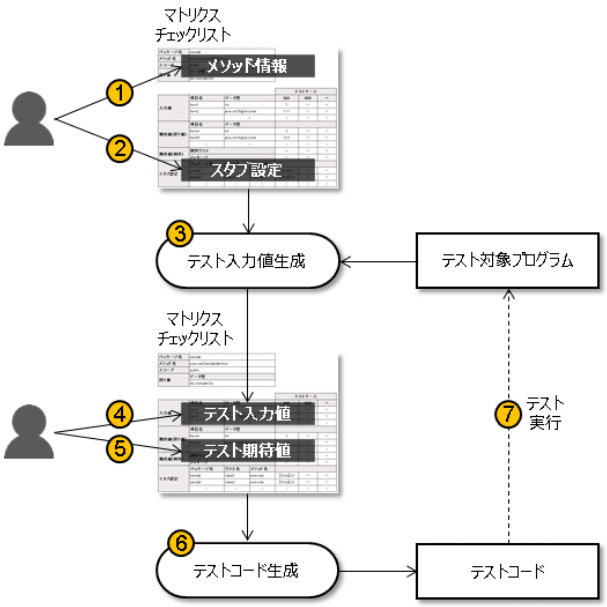

図 2: テスト自動化技術による単体テストのフロー概要

生成されたテスト入力値を確認し必要に応じてテスト入力値を追加し (図 2-④)，テスト入力値に対するテスト期待値を定める (図 2-⑤).

(2) テストコードの自動生成：マトリクスチェックリストからテストコードが生成される (図 2-⑥).

(3) テストの実行：生成されたテストコードを実行する (図 2-⑦).

4 おわりに

マトリクスチェックリストに基づくソフトウェアの単体テストを効率化するために，テスト自動化技術の開発・適用によるテスト方法を提案した．

今後は大規模エンタープライズアプリケーションの開発を中心に提案方法を適用しブラッシュアップを図る予定である．

参考文献

[1] 丹野治門，倉林利行，張 暁晶，伊山宗吉，安達 悠，岩田真治，切貫 弘之 ： テスト入力値生成技術の研究動向，コンピュータソフトウェア，Vol.34，No.3，pp.121-147(2017).

[2] Păsăreanu, C. and Rungta, N. : Symbolic PathFinder: Symbolic Execution of Java Bytecode, *Proc. IEEE/ACM International Conference on Automated Software Engineering(ASE 2010)*, pp.179-180(2010).

[3] Ohbayashi, H., Kanuka, H. and Okamoto, C. : A Preprocessing Method of Test Input Generation by Symbolic Execution for Enterprise Application, *Proc. 25th Asia-Pacific Software Engineering Conference(APSEC 2018)*, pp.717-718(2018).