6L-4

back-to-back テストを実行するテストツールの試作と評価

内海 武之　高木 智彦　八重樫 理人　古川 善吾
香川大学工学部

back-to-back テストは、大量のテストケースを生成し実行できるので高いソフトウェア信頼性を実現する有効な手法として考えられるものの、現場で広く活用されているとはいえない。本稿では、独自に検討した back-to-back テストの方法に基づいてテストツールを試作し、その結果と課題についてまとめる。

1. はじめに

要求されるソフトウェアの信頼性を実現する上で、テストは重要な役割を担っている[1]。ソフトウェアテストには様々な手法が存在し、そのひとつに back-to-back テストがある。

back-to-back テストは、テスト対象ソフトウェアの別バージョンを用いてテストデータに対する期待出力を自動的に導出する。言い換えれば、テスト対象ソフトウェアとその別バージョンに同じテストデータを適用し、その結果得られた出力を双方比較することによって、欠陥の有無を検出する手法である。ここでいう別バージョンとは、たとえばテスト対象ソフトウェアのプロトタイプや以前リリースしたバージョンのことである。これによりテストケース（テストデータと期待出力の組み合わせ）を自動的に生成し実行することが可能になるので、大量のテストケースを用いて高い網羅率を実現することができる。特に、高い信頼性が要求される分野で従来から利用されており、その有効性が認識されているものの、必ずしも利用が進んでいるわけではない。

そこで本研究では，back-to-back テストのためのツール（テストドライバ）を作成する手法について検討し、その試作を行う。そして back-to-back テストを実現する上での課題について明らかにする。

2. テストドライバの試作

2.1 back-to-back テストのためのフレームワーク

本研究では、まず back-to-back テストを実行するためのフレームワークを検討した。これの特徴は、SVM (Support Vector Machine) [2]を利用してテストの実行結果を自動的に判断する機能を備えている点である。概要を図 1 に示す。本研究における back-to-back テストの実施手順は、以下の 4 つのステップから構成される。

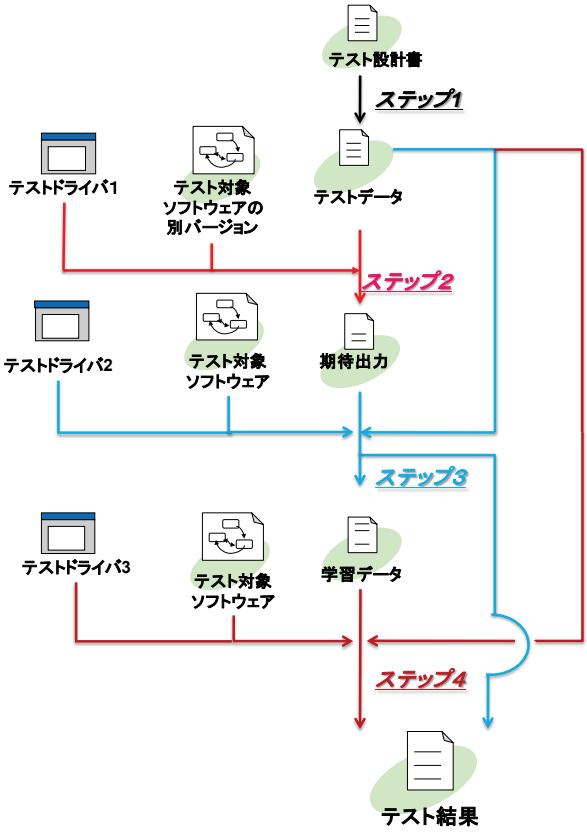

図 1. 本研究のフレームワークの概要

[ステップ1] テスト設計書に基づいて任意のテストデータ設計技法によりテストデータを作成する。テスト設計書とはテスト計画に基づいてテストケースの設計指針をまとめた文書である。

[ステップ2] テストドライバ 1 を用いてテスト対象ソフトウェアの別バージョンにテストデータを自動入力し期待出力の生成を行う。

[ステップ3] テストドライバ 2 を用いてテストデータをテスト対象ソフトウェアに自動入力し、その出力と期待出力を機械的に比較する。双方の出力内容に違いが検出された場合は、テスト対象ソフトウェアの欠陥に起因している可能性があるので、テスト担当者が出力結果を目視することにより欠陥か否かを判断し、その判断結果をテストドライバ 2 に入力する。テストドラ

Experimental Development and Evaluation of Test Tools for Back-to-back Testing

Takeshi Utsumi, Tomohiko Takagi, Rihito Yaegashi, Zengo Furukawa
・Faculty of Engineering, Kagawa University

イバ 2 は、出力内容の違い、およびテスト担当者による判断結果に基づいて、SVM に必要な学習データを生成する。すべてのテストデータの入力が完了すれば、テストドライバ 2 はテスト結果を出力する。テスト結果には検出した出力の違いに関する各種情報が含まれており、テスト担当者が後でテスト結果を確認したり、デバッグを行う際に役立てることができる。学習データを十分用意できた時点で、ステップ 4 に移行する。

[ステップ4] テストドライバ 3 を用いてテスト対象ソフトウェアに対してテストデータを自動入力し、その出力と期待出力を機械的に比較する。双方の出力内容に違いが検出された場合は、その違いが欠陥に起因するものか否かを SVM を用いて自動的に判定する。すべてのテストデータの入力が完了すれば、テストドライバ 3 はテスト結果を出力する。ステップ 3 との違いは、テストの実行結果の判定を人手で行うか、自動で行うかである。

2.2 テスト結果の自動判定方法

back-to-back テストの核心部分は、テスト対象ソフトウェアの出力を、その別バージョンの出力と比較するところにある。これによって、通常コストのかかる作業である期待出力の作成を自動化することができる。ただし、機械的に比較しただけでは、欠陥に由来しない些細な違いまでも不具合として検出してしまうという問題があることが知られている。これを解決するために本研究では SVM を導入した。

ステップ 3 において出力内容に差異が検出されたとき、テストドライバ 2 はその差異から特徴語を抽出し、その特徴語の出現頻度およびテスト担当者の判断結果を用いて学習データを生成する。たとえば、テスト対象ソフトウェアの出力に、本来含まれるべき「a」という文字が含まれていない箇所が 3 つある場合、まず「-a」という特徴語が識別される。ここで「-」は「含まれていない」という意味である。そして識別された特徴語に対して ID（たとえば 1）が付され、1:3 という学習データの一部が生成される。そして、もしこれが欠陥に由来するものであれば欠陥のクラスに対応するラベルが付され、欠陥でない場合は非欠陥のクラスに対応するラベルが付される。このようにして生成された学習データに基づいて、欠陥に由来する出力とそうでない出力のパターンを明らかにし、ステップ 4 においてテスト結果の判定に役立てる。

3. 実験

本研究では、前節にしたがって back-to-back テストを実行するテストドライバを試作し、これが機能することを確認するための予備的な実験を行った。用いた例題は、コマンドプロンプト上で動作する簡単なリバーシのプログラム（およそ 200LOC の C プログラム）である。ミューテーションテスト法で使用される定数置き換えにより、40 個のテスト対象ソフトウェアを用意した。それらのうち、(a) 20 個は欠陥を含むものであり、(b) 残り 20 個はオリジナルのバージョンとの差異はあるが欠陥を含まないものである。

実験では、まずステップ 1 として、プログラムの主要な機能を網羅するテストデータを手作業によって設計した。次にステップ 2 として、テストドライバ 1 を用いてオリジナルのリバーシのプログラムから期待出力を生成した。そしてステップ 3 として、(a)のうちの 10 個、および(b)のうちの 10 個に対して、テストドライバ 2 を用いたテストを実施した。その結果、(a)の 10 個すべてを欠陥があるものとして、また(b)の 10 個すべてを欠陥がないものとしてテスト担当者が容易に識別することができ、さらにそれらをふまえた学習データを生成できることを確認した。最後にステップ 4 として、ステップ 3 でテストを実行していない残りの 20 個（すなわち(a)の残り 10 個と(b)の残り 10 個）に対して、テストドライバ 3 を用いた自動テストを実行した。その結果、(a)の 4 個と(b)の 10 個については正しくテスト結果の判定を行うことができた。

4. おわりに

本研究では back-to-back テストのためのテストドライバについて検討した。そして試作したテストドライバを用いて予備実験を行った結果、自動的にテストケースを生成できることが分かった。また、SVM を用いた自動的なテスト結果の判定を試行したところ、十分な結果を得ることができなかった。その原因は、学習データを生成する際の特徴語を識別する機能が不十分だったことにあることが分かった。今後の課題として、学習データ作成方法の考察、およびテストドライバの改良を行い、テスト結果判定の精度向上を図っていく。さらに、テストドライバの作成を容易に行うことができるようにライブラリやフレームワークの整備を進めていく予定である。

参考文献

[1] R.V. Binder, "Testing Object-Oriented Systems", Addison Wesley, 1999.

[2] T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Scholkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.