ソフトウェアテスト自動化を改善する為のテスト手法

A Testing Method to Improve the Automation of Software Testing

八戸文仁　　伊藤恵

Fumihito Hachinohe　Kei Itou

公立はこだて未来大学

Future University-Hakodate

1　はじめに

本研究の目的は、従来のソフトウェアの単体テスト自動化の方法では解決することができない点を改善することである。現状では、自動でテストを行う為の準備段階にコストがかかる。また、複雑化したコードに対して自動でテストを行うことが困難である。この2点を改善することが本研究の目的である。

2　背景

最近、ソフトウェアテストを重視した開発プロセスも提案されてきている。例として、eXtreme Programmingやクリーンルーム開発法があげられる [5, 6, 7]。さらに、アジャイル開発プロセスの中にもソフトウェアテストを重視したものがある [7]。また、ソフトウェアテストの自動化に関する研究もなされており [2]、ソフトウェアテストのプロセスの1つである単体テストに対して有用であるモック・オブジェクトの自動生成に関する研究もされてきている [1]。

しかし、これまでの研究、特にソフトウェアの単体テスト自動化に関する研究において問題視されていることが3点ある。まず、全てのソフトウェアに対して1つの自動テストを適用することができないことである。次に挙げられるのは、自動テストの準備段階における時間や費用といったコストが大きいことである。最後に、様々な外部モジュールを利用するため複雑化してしまったソースコードに対して自動テストの適用ができないことである。このようなことから、開発現場でソフトウェアの単体テスト自動化を実際に導入できない場合がある。

また、近年、ソフトウェアの開発現場では、顧客から大規模かつ複雑な機能を要求されることが多い。さらに、短期間での開発を迫られることも少なくない。こうした現状において、ウォーターフォール型に分類される開発手法を用いている開発現場では、要求定義や設計、実装の段階にコストがかかり、ソフトウェアテストに割くコストが限られてくる。しかし、限られたコストの中でも高品質なソフトウェアを提供しなければならない。

3　研究概要

本研究は、従来のソフトウェア自動化で問題とされている2点を改善することを目的としている。問題点の1つとして挙げられる自動テストの準備段階におけるコストを軽減することが1つめの目標である。また、複雑化したソースコードに対してのテストを自動化できない点を改善することが2つめの目標である。

そこで、テスト仕様情報を厳密に形式化された情報としてソースコードを書いたプログラマが記述する。実際は、Fig. 1 のような XML として記述する。この情報を元に xUnit で使用できるテストコードを自動生成し、この自動生成されたテストコードを用いてプログラマ、もしくはテスターが単体テストを行う。自動生成されるテストコードの例を Fig. 3 として示す。この手法を用いることで、テスト仕様情報をあらかじめ定められた形式で記述するのみでテストコードを自動生成できるので、自動テストの準備段階におけるコストの軽減が可能である。また、ソースコードを書いたプログラマ自身がテスト仕様情報を XML として記述することで仕様記述に費やす時間を短縮することができる。但し、ソースコードを書いたプログラマ以外の人がテスト項目を作成することが望ましい。この一連の流れを Fig. 4 として示す。

また、テストをしたいソースコードが外部に依存する部分をモック・オブジェクトとして自動生成し、これをテストコードが使用する。このことにより、複雑に外部モジュールへ依存するソースコードに対しても完全な単体テストを行うことができる。

```
<class name="LabBean">
  <publicMethod>
    <method name="setLabNumber">
      <param type="class" class="String">
        <restriction maxlength="3" minlength="3" spe
      </param>
```

Fig. 1　記述する XML の例（一部）

```
public class LabBean  {
    public void setLabNumber(String param) {
        this.labNumber = param;
    }

    public String getLabNumber() {
```

Fig. 2 Fig. 1 の元になったソースコード（一部）

```
public class LabBeanTest extends TestCase {
    LabBean labBean;

    public void testSetLabNumber() {
        String stringMock = new Stirng("xxx");
```

Fig. 3 自動生成されたテストコード（一部）

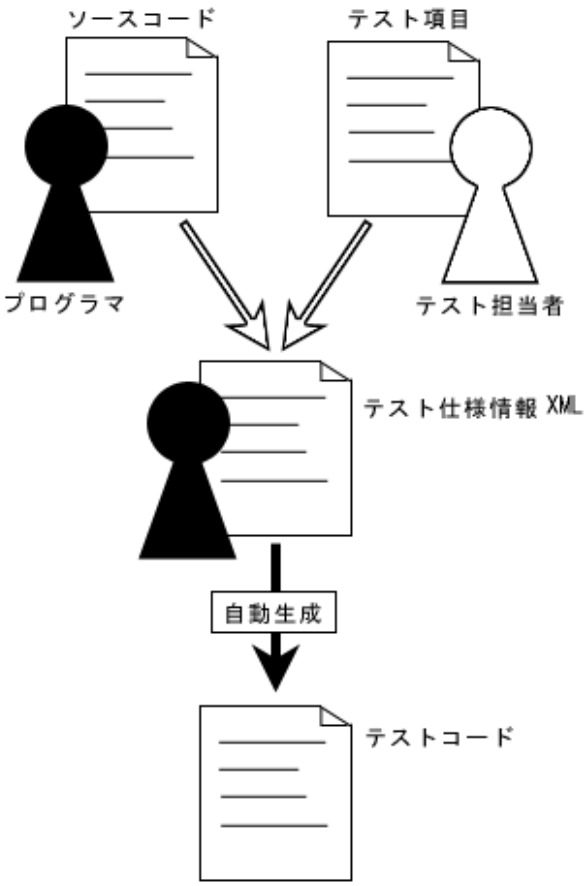

Fig. 4 単体テストの一連の流れ

4 評価方法

本研究が提案するテスト自動化手法によるソフトウェアの単体テストと従来の xUnit による単体テストとを比較する。この比較から、そのテスト網羅度やバグの発見率、テストに費やす時間などのコストの違いを調査する。また、この手法で自動生成されるテストコードを用いて正常にテストを行うことができるかを検査する。さらに、本研究が提案するテスト自動化手法を用いて作成されたソフトウェアが要求された機能を全て満たしていることを確認する。

5 今後の課題

この手法で作成されたソフトウェアの評価方法の吟味をし、それに基づいて評価を行う。この評価からさらなる改善点の考察を行う。発展的な課題として、テストコードの生成を自動化するのみではなく、テスト実行まで自動化する方法の提案とその実践、評価がある。また、テスト仕様情報を XML を記述する際に有用となるツールの開発、あるいは、ソースコードからテスト仕様情報 XML を自動生成する方法の提案も課題である。さらに、この手法を種々の開発手法やプログラミング言語にも適用できるように一般化を行う必要がある。

参考文献

[1] David Saff and Michael D. Ernst, Automatic mock object creation for test factoring, ACM SIGPLAN/SIGSOFT Workshop on Program Analysis for Software Tools and Engineering (PASTE'04), pp.49-51, 2004.

[2] David Saff and Shay Artzi and Jeff H. Perkins and Michael D. Ernst, Automatic test factoring for Java, ASE 2005: Proceedings of the 20th Annual International Conference on Automated Software Engineering, pp.114-123, 2005.

[3] 三浦公寿, XP における WEB テスト自動化, 2004, はこだて未来大学卒業論文.

[4] 原田真幸, Web アプリケーションにおけるテスト技法とプログラミングに関する研究, はこだて未来大学卒業論文.

[5] 日本 XP ユーザグループ, eXtreme Programming テスト技法 xUnit ではじめる実践 XP プログラミング, 翔泳社.

[6] Stephen R. Schach, OBJECT-ORIENTED & CLASSICAL SOFTWARE ENGINEERING, 2005, The McGraw-Hill.

[7] 玉井哲雄, ソフトウェア工学の基礎, 2004, 岩波書店.