様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２９　年　　６　月　２１　日現在

機関番号：　６２６１５
研究種目：　若手研究(B)
研究期間：　2014～2016
課題番号：　２６７３００４３

研究課題名（和文）Webアプリケーションのテストにおける正確で実用的な評価指標及び改善手法の確立

研究課題名（英文）Development of Precise and Practicable Evaluation Criteria and Improvement Method for Web Application Testing

研究代表者

坂本　一憲（Sakamoto, Kazunori）

国立情報学研究所・大学共同利用機関等の部局等・特任助教

研究者番号：６０６０９１３９

交付決定額（研究期間全体）：（直接経費）　　2,800,000円

研究成果の概要（和文）：本研究では、Webアプリケーション（Webアプリ）の仕様変更頻度が高く、Webアプリのテストコードの作成および保守コストが高いという問題の解決を目的とした。本目的を達成するために、Webアプリのテストコードの保守性を向上できるページオブジェクトデザインパターンに着目して、同パターンの適用を支援するための内部ドメイン特化言語および同言語の利用を支援するテスティングフレームワークDePoTを開発した。DePoTの評価実験を行ったところ、テストコードの作成および保守コストを削減できることが分かった。さらに、応用研究として、テスト入力の自動生成やWebページから情報を抽出する技術の開発に成功した。

研究成果の概要（英文）：The purpose of this study is to tackle the problem where Web applications frequently change and the development and maintenance costs of Web application tests are high. To achieve the purpose, we found that the PageObjects design pattern could improve the maintainability of Web application tests, and then, we developed a novel Web testing framework named DePoT including a new internal domain specific language for helping developers to employ the PageObjects design pattern. We evaluated DePoT and found that DePoT could reduce the development and maintenance costs of Web application tests. In addition, we successfully developed a new test input generation techniques named Feedback-controlled Random Test Generation and a new extraction method for Web pages as application studies.

研究分野：　ソフトウェア工学

キーワード：　ソフトウェアテスト　テストカバレッジ　テストオラクル

１．研究開始当初の背景

背景：Web アプリケーション（Web アプリ）の欠陥を検出するため、開発者は多数のテストケースから成るテストスイートを作成して、テストを実行する。テストスイートの欠陥を検出する能力（欠陥検出能力）を評価する指標（評価指標）と作成コストを考慮して作成を終了する。また、テストスイートが不十分な場合、テストスイートを改善する手法（改善手法）を用いて改善する。

問題：欠陥の検出において、[要件 1] 欠陥検出能力の正確な評価が必要である。テストスイートを構成するテストケースは、テスト対象のプログラムを実行する入力（入力）と、プログラムからの出力を監視して欠陥の有無を判定するテストオラクル（監視処理）から成り、両方の評価が必要である。また、実用上、[要件 2] 評価の計算時間をテスト実行の十倍程度に抑える必要がある。テストスイートが不十分な場合は、[要件 3] 重大な欠陥を検出可能な改善手法が必要である。しかし、要件 1-3 を満たす Web アプリの正確で実用的な評価指標と有用な改善手法が存在しないため、開発者は欠陥検出能力が不十分なテストスイートを十分と誤認し、十分な改善を行わない問題がある。その結果、開発者が重大な欠陥を見過ごし、利用者に社会的不利益を与える。

既存の評価指標：Web アプリで利用可能な評価指標に、テスト網羅性とミューテーション解析がある。テスト網羅性はテスト時に実行したプログラム要素の割合を示し、[問題 1] 入力のみを評価するため不正確な指標である[a1]。ミューテーション解析は機械的に欠陥を埋め込み、埋め込んだ欠陥の検出数で評価する。ミューテーション解析は正確だが、小規模 Web アプリでもテスト実行の数百倍以上の計算時間を要し、実適用が困難である[a2]。その他に、監視処理も評価可能なテスト網羅性が提案されているが、適用範囲に制約があり Web アプリに適用できない[a3]。

既存の改善手法：Web アプリにおいて、入力と監視処理の生成手法が両方存在しており、入力生成は改善に有用である[a4]。一方、監視処理の生成は不変条件を用いているが、[問題 3] 不変条件は HTML 文章中の不変な性質のみを監視するため、静的部分の軽微な欠陥しか検出できない[a5]。

＜引用文献＞

[a1] M. Staats et al., "Programs, Tests, and Oracles: The Foundations of Testing Revisited", ICSE 2011, pp.391-400.
[a2] J. H. Andrews et al., "Is Mutation an Appropriate Tool for Testing Experiments?", ICSE 2005, pp.402-411.
[a3] M. Whalen et al., "Observable Modified Condition/Decision Coverage", ICSE 2013, pp.102-111.
[a4] S. Thummalapenta et al., "Guided test generation for web applications", ICSE 2013, pp.162-171.
[a5] A. Mesbah et al., "Invariant-Based Automatic Testing of Modern Web Applications", IEEE TSE, pp.35-53, 2012.

２．研究の目的

本研究では、以下で述べる A-C の 3 項目を実現することを目的としていた。

(A) Web アプリにおいて入力と監視処理の組合せを評価する正確で実用的な評価指標の確立

テスト網羅性は実行したプログラム要素に着目して入力を、動的部分の監視範囲は監視した動的部分に着目して監視処理を評価する。そこで、入力と監視処理の組合せを評価する、テスト網羅性よりも正確な評価指標を確立する。ミューテーション解析では、埋め込む欠陥数と同じ回数のテスト実行が必要だが、提案指標では一回だけで、テスト実行の数倍程度の計算時間である。

(B) Web アプリ中の動的部分に対する監視処理を生成する改善手法の確立

利用者にとって、静的部分（例：「ようこそ」の表示）で生じる欠陥よりも、動的部分（例：所持金の表示）で生じる欠陥の方が重大である。そこで、提案指標の計算に必要な動的部分の抽出を応用して、抽出した動的部分に対する監視処理の生成手法を確立する。提案手法は、Web アプリ中に存在する全ての動的部分を監視するように処理を追記して、既存テストスイートを改善する。

(C) 実 Web アプリ開発における有効性の評価

実 Web アプリ開発において次の三点を確認して、有効性を評価する。1) 提案指標がテスト網羅性よりも正確である点。2) 提案指標がミューテーション解析よりも少ない、受け入れ可能な計算時間で計算できる点。3) 改善手法が動的部分の重大欠陥を検出可能な監視処理を生成できる点。

３．研究の方法

(A) Web アプリにおいて入力と監視処理の組合せを評価する正確で実用的な評価指標の確立

当初はテストオラクルの品質測定技術を改良することで[b1]、入力と監視処理の組み合わせを評価する新しい技術の開発に取り組む予定だったが、両者を統合的に評価する指標の確立が難しかった上、単純に、テスト網羅性とテストオラクルの品質測定技術を組み合わせることで、当初の目標を達成することができた。そのため、項目 A については、新たに研究開発を行う必要がないと判断して、項目 B について注力することにした。

(B) Web アプリ中の動的部分に対する監視処理を生成する改善手法の確立

Web アプリは従来のソフトウェアと比較して、仕様変更の頻度が高く、テストコードの保守コストが高いという問題がある。そこで、上記問題を緩和しながら Web アプリの動的部分に対する監視処理を実装する上で、保守コストの削減効果の高いページオブジェクトデザインパターンに着目した。従来は、開発者が人手でページオブジェクトデザインパターンをテストコードに適用しながら、テストコードを記述する必要があった。しかし、ページオブジェクトデザインパターンを適用したテストコード中には、くり返し現れる表現が存在することに気付き、ページオブジェクトデザインパターンを適用したテストコードを記述するための内部ドメイン特化言語を提案することにした[c1]。

前述の提案は、任意の Web アプリを対象に適用することを目指すこととした。一方、EC サイトの商品説明の Web ページなど、デザインは同じで内容の異なる Web ページ群が存在する。前述の提案について研究を進める中で、上記のような Web ページ群を対象とすることで、Web ページ群の相違点をテストオラクルとして利用できる可能性に気付いた。そこで、テストオラクルの保守が必要なケースを減らして、結果的に、テストコードの保守コストを削減するために、相違点をテストオラクルとする手法を提案することにした。また、上記提案が、Web アプリのテストのみならず、Web ページから情報を抽出する技術に応用できることに気付き、実用化のための研究開発を行うこととした[c3, c4]。

以上の提案は、テストオラクルの改善手法を扱っていたが、研究を進める中で、テストの入力を自動生成する手法についてヒントを得た。そこで、Feedback-controlled Random Test Generation と名付けた新しいテスト入力の自動生成手法を提案することにした[c2]。

(C) 実 Web アプリ開発における有効性の評価

当初は民間企業と共同研究を行い、製品として利用されている Web アプリを対象に実験を行う予定であった。しかし、民間企業のニーズを満たすことができず、共同研究契約の締結ができなかった。そこで、オープンソースソフトウェアを対象として、上記提案手法の評価実験を行い、それぞれの有効性を検証することにした。

＜引用文献＞

[b1] K. Sakamoto et al., "POGen: A Test Code Generator Based on Template Variable Coverage in Gray-Box Integration Testing for Web Applications", The 16th International Conference on Fundamental Approaches to Software Engineering, pp.343-358, 2013.

４．研究成果

４．１．DePoT：Web アプリケーションテストにおけるテストコード自動生成テスティングフレームワーク

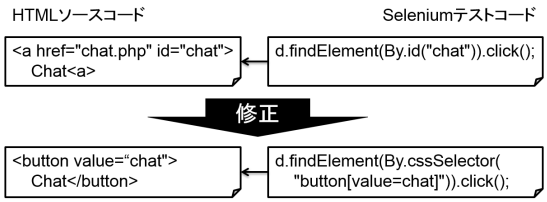

図 1. Web アプリの仕様変更例 [c1]

Web アプリは従来のソフトウェアとして仕様変更が頻繁に起こる。そのため、Web アプリのテストコードは保守コストが高くなりがちである。図 1 で仕様変更の例を示す。図 1 では、Chat という文言のリンクがボタンに変化しており、そのため、テストコードも同様の修正が必要となっている。

ページオブジェクトデザインパターンを適用したテストコードの記述を支援するために、ページオブジェクトデザインパターンの適用を前提とした、新しい内部ドメイン特化言語（内部 DSL）を開発して、内部 DSL を記述するためのテスティングフレームワーク DePoT を開発した。

DePoT は、特徴 1）既存の Web アプリ向けテスティングフレームワーク Selenium が提供する機能を活用できること、特徴 2）文脈依存／非依存なアサーションを分離できること、特徴 3）テストケースをメソッドチェーンのみで記述できることの 3 つの特徴を備えている。

DePoT の有用性を検証するために、RQ1）DePoT はテストコードの作成コストを削減できるか？RQ2）DePoT はテストコードの修正コストを削減できるか？という 2 つの研究課題を掲げた。実験参加者 6 名の協力を得て、実験 1）テストコードの内容を理解する実験、実験 2）Web アプリの仕様変更に対してテストコードを修正する実験、実験 3）テストコードを作成する実験の 3 つの実験を実施した。

その結果、実験 1 において、テストコードの内容の理解に必要な時間は、従来手法では平均 3 分 42 秒であったに対して、DePoT では平均 2 分 35 秒であった。実験 2 において、仕様変更に対するテストコードの修正に要した時間が、従来手法では平均 5 分 40 秒であったのに対して、DePoT では平均 2 分 18 秒であった。実験 3 において、作成したテストコードの行数を測定したところ、従来手法では 251 行であったのに対して、DePoT では 209 行であった。以上から、DePoT を利用することで、テストコードの理解容易性を向上させ、テストコードの修正コストを削減して（RQ2）、テストコードの作成コストを削減できる（RQ1）ことが分かった。

表 1. 従来手法と提案手法の比較結果

命令網羅率(ステートメント力バレッジ)

ソフトウェア	実行時間(秒)	従来手法		提案手法		
		網羅率(中央値)	四分位範囲	網羅率(中央値)	四分位範囲	改善率
collections	60	24.6	3.0	44.0	2.3	78%
	3600	45.5	16.8	70.6	0.8	55%
lang	60	42.2	12.4	58.5	6.4	38%
	3600	62.4	3.4	76.6	0.6	22%
guava	60	18.7	4.7	23.1	3.4	23%
	3600	27.7	9.6	40.3	0.6	45%
math	60	14.3	5.5	21.2	5.7	48%
	3600	25.3	14.0	50.4	1.0	98%
codec	60	73.1	1.7	75.2	0.6	2%
	3600	80.5	0.5	81.9	0.3	1%
gson	60	49.5	3.2	55.8	1.2	12%
	3600	60.6	1.2	63.3	0.5	4%
h2	60	17.7	4.8	16.3	9.7	−
	3600	27.2	5.9	28.2	0.7	−
jetty	60	18.5	2.5	20.2	1.2	8%
	3600	28.5	1.2	28.5	1.7	−

分岐網羅率(ブランチ力バレッジ)

ソフトウェア	実行時間(秒)	従来手法		提案手法		
		網羅率(中央値)	四分位範囲	網羅率(中央値)	四分位範囲	改善率
collections	60	6.0	1.5	14.1	1.9	136%
	3600	16.9	10.8	44.4	0.8	162%
lang	60	16.1	6.6	35.1	6.8	118%
	3600	33.0	5.3	58.8	0.4	78%
guava	60	6.7	2.0	10.0	2.5	50%
	3600	12.3	7.6	27.5	0.6	123%
math	60	5.6	2.9	9.8	3.7	75%
	3600	11.4	7.4	34.7	1.1	204%
codec	60	45.9	2.4	51.1	1.1	11%
	3600	61.4	0.7	65.1	0.8	6%
gson	60	28.3	4.4	36.0	0.9	27%
	3600	41.9	1.2	44.6	0.9	6%
h2	60	4.8	2.6	5.0	3.3	−
	3600	10.6	3.4	12.3	0.3	−
jetty	60	2.7	1.1	3.4	1.0	29%
	3600	10.8	0.9	12.2	0.4	12%

４．２．Feedback-controlled Random Test Generation

従来のテスト入力を自動生成する手法として、ランダム生成手法が有名であった。しかし、ランダム生成手法では、リスト構造や木構造のオブジェクトなど、複雑なオブジェクトを生成することが困難であった。そこで、Pacheco らは、ランダム生成手法で生成した入力を使ってテスト対象のメソッドを実行して、得られた出力結果を蓄積し、その後、出力結果を別のテスト対象のメソッドに入力する Feedback Random Test Generation 手法を提案した。同手法はメソッドの出力結果を活用することで、従来のランダム生成手法では生成できなかったオブジェクトの生成を可能にして、テスト網羅性の高い入力の生成を実現した。

しかし、Feedback Random Test Generation 手法では、メソッドの出力結果を入力に利用するというフィードバックループが、入力の生成を過剰に指向付けることで、生成されるテスト入力が偏る傾向があることを発見した。このことが、生成される入力のテスト網羅性が低く、また、生成を試行する度に、得られる入力の質が変化しやすいという問題がある。そこで、出力結果を蓄積するプールを複数用意して、さらに、1) 利用するプールを選択すること、2) 空のプールを追加すること、3) 既存のプールを削除することの 3 つの操作を定義して、生成される入力が偏らないように、プールを操作する Feedback-controlled Random Test Generation 手法を提案した。

提案手法の有効性を評価するために、8 種類のオープンソースソフトウェアを対象として、従来手法と提案手法で比較実験を行った。実験では、60 秒間と 3600 秒間の 2 種類の設定で、それぞれ 10 回ずつ入力を生成して、命令網羅率と分岐網羅率の中央値および四分位範囲を測定した。その結果、表 1 のような結果が得られた。提案手法のほうが、網羅率の中央値が向上していることが分かる。特に、命令網羅率では最大 98%、分岐網羅率では最大 204%の改善を実現した。

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計　1　件）
[c1] 青井翔平，坂本一憲，鷲崎弘宜，深澤良彰，"DePoT：Web アプリケーションテストにおけるテストコード自動生成テスティングフレームワーク,"情報処理学会論文誌，56(3)，pp.835-846，2015 年 3 月．[査読有り]

〔学会発表〕（計　1　件）
[c2] Kohsuke Yatoh，Kazunori Sakamoto，Fuyuki Ishikawa and Shinichi Honiden，"Feedback-controlled Random Test Generation," International Symposium on Software Testing and Analysis (ISSTA 2015)，pp.316-326，2015 年 7 月，Maryland，US．[査読有り、トップカンファレンス(Core Rank A)、Best Artifact Award 受賞]

〔その他〕
[c3] 産学連携で研究成果を社会実装へ/アーキテクチャ科学研究系 坂本一憲助教の web 情報抽出技術
http://www.nii.ac.jp/news/release/2017/0208-1.html

[c4] 国立情報学研究所の web 情報抽出技術に関する研究開発に参画して研究成果を事業化
http://rooter.jp/news/2017/nii_release.html

６．研究組織
(1)研究代表者
坂本　一憲　（SAKAMOTO，Kazunori）
国立情報学研究所・アーキテクチャ科学研究系・助教
研究者番号：60609139