R-1

Web アプリケーションのためのサーバを必要としない テスト支援ツールの研究

橋浦研究室　　1145431　茂木　康汰

1. はじめに

近年，Web アプリケーションが急速に広まっている．加えて，近年では，シングルページアプリケーションが多く用いられるようになってきている．それに，Web アプリケーションは，一般のアプリケーションと比較して，機能の追加などの仕様変更が多いという特徴がある[1]．よって，回帰テストの重要性は，高まり続けている．しかし，回帰テストを行うための準備はとても手間がかかる．

2. 研究目的

本研究の目的は，回帰テストの準備に必要なテストコードを作成する作業，サーバの環境を整える作業に着目し，その支援を行うことを目的とし，テストコードを自動生成する手法とサーバと通信を行わない手法の提案をする．そして，提案した手法を実現するツールを開発し，開発したツールを用いて実験を通し，提案した手法の有効性を検証する．

3. 提案手法

本研究の提案手法は，Web アプリケーションの回帰テストを実施する際，サーバと通信を行う部分をモックに差し替えることで，サーバと通信を行わない環境を実現させ，自動で，テストコードを作成・実行してテストするものである．それ故に，モックが必要な情報，テストコードを作成するための情報が必要となり，自動で行う必要がある．

4. ツールの実現方法

提案手法を実現させるためには，以下の 4 つの機能が必要である．

ⅰ）`XMLHttpRequest` の状態を保存する
ⅱ）テストコードを作成する
ⅲ）`XMLHttpRequest` のモック
ⅳ）テストコードを実行する

上記の機能は，Chrome 拡張機能を利用して実現させる．ⅰ），ⅲ）は，サーバと通信を行わないための機能である．具体的な処理手順は図 1 に示す．まず，`XMLHttpRequest` とサーバの通信内容をリクエストとレスポンスの組み合わせでファイルに保存する．その後，`XMLHttpRequest` をモックに差し替え，モックはファイルの情報を利用し，同一の振る舞いをする．従って，他の機能からは `XMLHttpRequest` のように扱うことができ，かつ，サーバと通信を行わなくて済む．よって，サーバの環境を整える必要もなくなる．

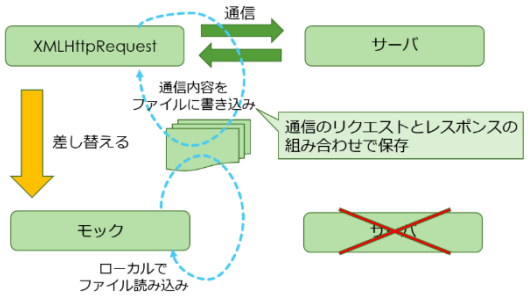

図 1　サーバを必要としない仕組み

5. 実験と考察

本研究で提案するツールが有効であるか検証するために，以下の 3 つの観点から，それぞれ検証を行った．

1）モックが `XMLHttpRequest` のような振る舞いであるか
2）自動で生成されたコードがどの程度効果があるか
3）回帰テストを支援できているか

実験の結果，一定のサイト，プログラムでツールが有効であることを確認することができたが，まだ汎用性が高くはないことがわかった．また，本研究ツールが回帰テストを支援できていることを確かめることができた．以下に各実験の詳細を述べる．

5.1. モックの振る舞い検証

実験対象は，文部科学省の Web サイトに掲載されている，東京都・埼玉県・千葉県・神奈

川県の私立大学の Web サイトの中で, jQuery を利用し，Ajax 通信を行っている 18 個のサイトとした.

テスト内容は以下の順序で行った.

A) 本研究ツールを利用せずに，テスト対象をブラウザでアクセスする
B) 本研究ツールの `XMLHttpRequest` のモック機能を，`XMLHttpRequest` の情報を削除した状態で，テスト対象をブラウザでアクセスする
C) 本研究ツールの `XMLHttpRequest` のモック機能を，`XMLHttpRequest` の情報を保存した状態で，テスト対象をブラウザでアクセスする

成否については，成功の場合を画面に変化がなく，コンソール画面でもエラーや `null`, `undefined` などが見られないものとし, 画面に変化や，Ajax 通信が失敗したと思われる内容が表示されている場合は，失敗と見なすこととした.

モックの振る舞い検証の結果を表 1 に示す. 約 4 割のサイトでは，`XMLHttpRequest` と同じ振る舞いをしていることが確認できた. 現在，本研究ツールは，一つの URL で，一つの情報を保存する仕組みになっているため, 8 個のサイトでは，複数の Ajax 通信に対応することができなかった. 3 個のサイトでは，レスポンス情報に `parse()`できない情報が入っているため，エラーが出てしまったと考えられる.

表 1　モックの振る舞い実験の結果

成功数	7サイト (約40%)		
失敗数	11サイト (約60%)	JSON.parse()エラー	3サイト
		Ajax通信のレスポンスエラー	8サイト

5.2. テストコードの効果検証

実験対象は，筆者が作成を行い，第三者が修正したプログラムとする. これは, Git で管理されているプログラムであり，Ajax 通信を行う機能が存在する commit(更新)で提案したツールを利用して, `XMLHttpRequest` の状態とテストコードを作成するために必要な情報を保存している. 修正後の commit に変え，`XMLHttpRequest` のモック機能とテストコードを自動で生成・実行する機能を利用する.

評価指標は，メソッド数，作成数，成功数，失敗数，エラー数で行う. メソッドをテストするコードの数量を作成数とする. 成功・失敗については表 2 に示す. この際，テストの結果が表示されなかったものはエラーとして扱った.

テストコードの効果検証の結果を表 3, 表 4 に示す. 2 つの commit で 7 割の成功を確認できた. バグが存在している状態で，テストの情報を保存し，第三者が修正したため，ツールはバグを見つけることができず，2 回目の修正でバグが修正されたため，バグではないのにバグであると表示したと考えられる.

表 2　テストコードの成功・失敗の定義

	バグがあると表示	バグが無いと表示	メソッドが修正されたと表示
バグが存在する	成功	失敗	失敗
バグが存在しない	失敗	成功	失敗
メソッドが修正された	失敗	失敗	成功

表 3　テストコードの効果実験の結果

メソッド数	作成数	成功数	失敗数	エラー数
6個	6個	5個	1個	0個
6個	6個	4個	2個	0個

表 4　実験結果の成功・失敗の個数

	バグがあると表示	バグが無いと表示	メソッドが修正されたと表示
バグが存在する	0個	1個	0個
バグが存在しない	2個	5個	0個
メソッドが修正された	0個	0個	4個

5.3. ツールの回帰テスト支援効果検証

実験対象は，著者が所属する研究室の学生 4 人に，5.2 節で利用したプログラムを対象に手動とツールを利用した場合で回帰テストを行った.

その手順を表 5 に示す.

表 5　ツールの回帰テスト支援効果検証の手順

本研究のツールを利用しない	本研究のツールを利用
①実験対象の環境を整える	Ⅰ. 実験対象の環境を整える
②被験者にテストコード作成	Ⅱ. ツールを利用して、テストの情報を保存
③mergeしたcommitに変更	Ⅲ. mergeしたcommitに変更
④サーバ関係の情報を削除	Ⅳ. サーバ関係の情報を削除
⑤被験者にサーバ環境を整えてもらう	
⑥テストコードを実行してもらう	Ⅴ. 本研究ツールを使ってテスト

評価方法は，回帰テストにかかった時間とテストコードの結果を比較する．①とⅠ，③とⅢ，④とⅣは同じ作業のため，時間の計測を行わない(表 5 の網掛け部分)．テストコードの結果に関しては，5.2 節と同じ評価方法で成否を判断し，比較する．

所要時間の結果を表 6，テストコードの結果を表 7 に示す．結果より，回帰テストにかかる時間を大幅に削減でき，テストコードの効果も良い結果を出せることが確認できた．加えて，メソッドの特徴で人間，ツールに差が生まれることが確かめられた(表 8)．メソッド B のような，修正がされておらず，単純な値を返す場合は差が生まれにくいと考えられる．メソッド D，E，F のようなメソッド修正がされており，表示などのレスポンスを返さないメソッドに関しては，ツールが人間より有効である部分である．これは，メソッド情報などを保存しているため，修正されているかの検知ができるためであると考えられる．メソッド A に関しては，人間・ツール共に失敗しているが，ツールによるテストコードを埋める位置が悪いためによるもので，他の commit ではツールのみ成功している．このメソッドはレスポンスが複雑なものなので，人間よりツールの方が得意だと考えられる．メソッド C に関しては，5.2 節で述べたように，既存のバグが入っていたため，ツールも人間も良い結果を出すことができなかった．しかし，ソースコードを読み正しいテストコードを作成できた人間もいたため，既存のバグが入っている場合は，人間の方がツールより良いと考えられる．

表 6　ツールの使用/不使用における所要時間の比較

	ツール不使用	ツール使用
被験者A	56分49秒	25秒
被験者B	59分30秒	28秒
被験者C	56分3秒	22秒
被験者D	35分28秒	18秒
平均値	51分58秒	23秒

表 7　ツールの回帰テスト支援効果検証のテストコードの効果の評価結果

	作成数	成功数	失数	エラー数
ツール	6個	4個	2個	0個
被験者A	6個	1個	5個	0個
被験者B	6個	2個	4個	0個
被験者C	6個	1個	5個	0個
被験者D	6個	1個	5個	0個

表 8　テストコードの効果のメソッドごとの結果

	メソッドA	メソッドB	メソッドC	メソッドD	メソッドE	メソッドF
ツール	×	●	×	●	●	●
被験者A	△	●	×	△	△	△
被験者B	×	●	●	△	△	△
被験者C	△	●	×	◆	◆	◆
被験者D	×	●	×	△	△	△

● 成功
× 失敗
◆ レスポンス検査は成功しているが、メソッドの修正検査ができていない
△ 結果のメッセージは一致してるが、テストの中身が間違っている

6. まとめと今後の課題

本研究では，回帰テストの準備に必要な，テストコードを作成する作業，サーバの環境を整える作業に着目し，サーバと通信を行わなくて済み，テストコードを自動で生成・実行する手法をツールに実現した．実験の結果によって，一部のサイトでは効果が見られたことから，回帰テストの準備に関して支援の効果が確認できた．

しかしながら，効果が見られないサイトも明らかになったため，さらに汎用性を上げる必要がある．サーバと通信を行わない機能に関しては,複数の Ajax 通信に対応することや，`XMLHttpRequest` 以外の Ajax 通信を行う Fetch API などに対応するなどの課題がある．テストコードを作成・実行する機能に関しては，jQuery などのフレームワーク対応などの課題がある．

参考文献

[1]Jazayeri, Mehdi, “Some Trends in Web Application Development,” International Conference on Software Engineering 2007 Future of Software Engineering, pp. 199-213, 2007-05.