2008－SE－161 (9)
2008／9／26

Webアプリケーションを対象とした回帰テスト支援ツール

今関雄人† 高田眞吾†

Webアプリケーションには，一般のアプリケーションと比較して，機能の変更が多いという特徴がある．このため，回帰テストが非常に重要である．Webアプリケーションの回帰テストを支援するツールとして，入出力を保存するものがある．これらのツールは，保存した入力を再現することにより，一度行ったテストを再実行することが可能である．また，再実行した際の出力と，保存した出力を自動的に比較することができる．しかしながら，これらのツールは，クライアントサイドの入出力，すなわち，HTTP リクエスト/レスポンスのみに着目しており，サーバサイドの入出力，すなわちデータベースやセッション変数に関しては扱うことができないという問題がある．そこで，本論文では，サーバサイドの入出力を考慮した，Webアプリケーションの回帰テスト支援ツールを提案する．本ツールを使用することにより，Webアプリケーションの回帰テストを自動的に行うことができる．

A Regression Test Supporting Tool for Web Application

Yuto Imazeki † and Shingo Takada†

The regression testing of web applications is very important because web applications have the feature that their requirements are frequently changed, compared with general applications. Some regression test support tools can save input and output of web applications, thereby re-executing a test by restoring saved input, and automatically comparing the output with the saved output. However, these tools focus only on client side information such as HTTP request/response, and do not handle server side information such as database and session variables (HTTP Session). In this paper, we propose a tool that supports regression testing of web application, that supports both server side and client side information. Our tool makes possible automatic regression testing of web applications.

1. 背　景

近年のインターネットの普及により，Webブラウザを使用してネットワーク越しにアクセスする，Webアプリケーションが急速に広まっている．Webアプリケーションは，近年ではエンタープライズシステム等にも広く用いられており，信頼性を確保するテスト手法の必要性が高まっている．

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，機能追加などの仕様変更が非常に多いという特徴がある．従来のデスクトップアプリケーションが数ヵ月，あるいは年単位でリリースされていたのに対して，Webアプリケーションは1日に数回更新されることも珍しくない[1]．そのため，仕様の変更により既存の機能が影響を受けていないかを確認する，回帰テストが非常に重要であると言える．

一般的に，回帰テストの研究では，実行するテストの数を最小限に抑えるもの[7]や，テストの実行順序に優先順位をつけるもの[5]が多く，テスト時間を削減することに主眼が置かれている．しかしながら，これらの研究は，既にテストケースが存在しているという前提で行われており，どのようにしてテストケースを用意するかについては考慮されていない．実際に回帰テストを行う際には，十分な数のテストケースを用意しなければならないため，どのようにしてテストケースを用意するかを考える必要がある．

回帰テストのテストケースを用意するには，一度行ったテストの入出力を保存しておき，保存した入力を使用してアプリケーションを再実行し，出力結果を保存したものと比較するのが効率的である．Webアプリケーションの入出力を保存し，回帰テストを支援するツールとして，Selenium[4]やJMeter[2]がある．これらのツールは，保存した入力を再現することにより，テストを再実行可能である．また，再実行した際の出力と，保存した出力を自動的に比較することができる．こららの機能により，一度実行したテストの入出力を蓄積し，再テストすることができる．これによ

† 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University

り，回帰テストを半自動的に行うことができる．

しかしながら，これらのツールは，クライアントサイドの入出力，すなわち，HTTP リクエストおよびレスポンス (出力 HTML) のみに着目しており，サーバサイドの入出力，すなわち データベースやセッション変数の入出力に関しては扱うことができないという問題がある．そこで，本論文では，サーバサイドの入出力を考慮した，Web アプリケーションの回帰テスト支援ツールを提案する．本ツールは，入力として，HTTP リクエスト，データベース，セッション変数，サーバ時刻を扱い，出力として，HTTP レスポンス，データベース，セッション変数を扱う．本ツールを使用することで，テストに使用した入出力を保存し，再実行を行うことができる．これにより，Web アプリケーションの回帰テストを自動化することができる．

本論文の構成は次の通りである．まず 2 章で既存の回帰テスト支援ツールの関連研究と，その問題点について述べる．3 章では提案する回帰テスト支援ツールについて述べ，4 章ではその実装について述べる．そして，5 章で提案ツールの適用例を示し，考察を行う．最後に，6 章で結論を述べる．

2. 関連研究

本節では Web アプリケーションの回帰テストを支援する関連研究と，その問題点について述べる．

2.1 Selenium

Selenium[4] は，Web ブラウザの操作を自動化することにより，Web アプリケーションのテストを行うツールである．

Selenium のテストケースは，ブラウザに表示されたページに対する操作 (フィールドへの入力や，ボタンのクリック) および，アサーションからなる．アサーションとは，操作の結果得られたページの検査を行う命令であり，回帰テストの際に結果を自動比較するのに使用される．例えば，ページのタイトルが指定した文字列と一致しているかどうかなどを，アサーションを用いて検査し，結果が正しいかどうかを確認する．

Selenium は，ブラウザを自動的に操作することにより，保存されたテストケースの入力を再現し，テストの再実行を行う．そして，操作の結果が正しいかどうかを，アサーションを用いて自動的に判定することができる．保存された複数のテストケースを一括で再実行することもできるため，回帰テストをほぼ自動的に行うことができる．

Selenium のテストケースは，HTML の table 形式を用いて，手動で記述する必要がある．しかし，Selenium IDE というツールを使用することにより，ブラウザに表示されたページに対するユーザの操作を自動的に記録し，テストケースとすることが可能である．ただし，記録できるのは操作のみであり，アサーションはテスト作成者が手動で付与する必要がある．

2.2 JMeter

JMeter[2] は，パフォーマンス測定，負荷テストなどの様々な機能を持つ汎用テストツールである．

JMeter では Web アプリケーションも扱うことができ，Web アプリケーションの入力として HTTP リクエスト，出力として HTTP レスポンスを扱う．HTTP レスポンスの検査は，Selenium 同様アサーションを用いる．アサーションでは，指定した単語が含まれているかどうかという基本的なもののほかに，HTML の文法が妥当であるかなども検査することができる．JMeter は 保存した HTTP リクエストを送信し，受信した HTTP レスポンスについて，アサーションを用いて結果の検査を行うことにより，回帰テストを自動的に行うことができる．

また，JMeter はテストケース生成のためのプロキシ機能を持ち，通常のブラウザを用いて対象 Web アプリケーションにアクセスするだけで，HTTP リクエストを保存し，テストケースとすることができる．このため，Selenium のように明示的に回帰テストのためのテストケースを作成する必要がなく，テストケースの生成が容易である．ただし，アサーションを別途手動で付け加える必要がある点は，Selenium と同様である．

2.3 問題点

Selenium や JMeter のような既存ツールを使用することで，Web アプリケーションに対し行ったテストの入力と出力を保存することができる．そして，保存したテストを再実行し，保存した結果と自動的に比較することができる．これらの機能により，回帰テストを半自動的に行うことができる．

しかしながら，既存ツールは，HTTP リクエストや HTTP レスポンス (出力 HTML) といった，クライアントサイドから見た入出力しか扱うことができないという問題がある．Web アプリケーションには，データベースやセッション変数を使用するものが非常に多く，そのような Web アプリケーションは，データベースの内容や，セッション変数の状態により，動作が変化する．例えば，商品をデータベースで管理しており，その商品の一覧を表示するページでは，商品の追加等により，データベースの内容が変わると，出力 HTML の内容も変化する．このようなアプリケー

ションを既存ツールで回帰テストする場合，テストを作成した時と，再テストする時で，データベースの内容が異なっていると，出力の HTML も変化してしまい，実行結果を正しく比較できないという問題がある．

また，データベースやセッション変数は，入力であると同時に，出力でもある．すなわち，データベースへの書き込みを行うようなアプリケーションをテストする際には，データベースへ正しく書き込まれていることを確認する必要がある．例えば，商品の注文情報を書き込み，注文完了画面を表示するようなページでは，出力 HTML の内容だけでなく，データベースに正しく注文情報が書き込まれていることも確認する必要がある．しかしながら，既存ツールでは，出力 HTML の内容しか確認することができない．

このように，既存ツールは，クライアントサイドから見た入出力のみを扱っており，サーバサイドの入出力に関しては扱うことができないという問題がある．

3. 提　　案

本論文では，サーバサイドの入出力を考慮した，Web アプリケーションの回帰テスト支援ツールを提案する．本ツールを使用することで，テストに使用したクライアントとサーバの両サイドの入出力を保存し，再実行を行うことができる．さらに，本ツールは再実行した結果を自動的に比較する機能を持つ．これらの機能により，Web アプリケーションの回帰テストを自動化することができる．

本ツールの全体像を図 1 に示す．本ツールはサーバに配置するツールであり，JMeter と同様，プロキシ機能を持つ．本ツールでは，クライアントと Web アプリケーションの間および，サーバ (データベースやセッション) と Web アプリケーションの間をフックすることにより，テストの保存および再実行を実現する．

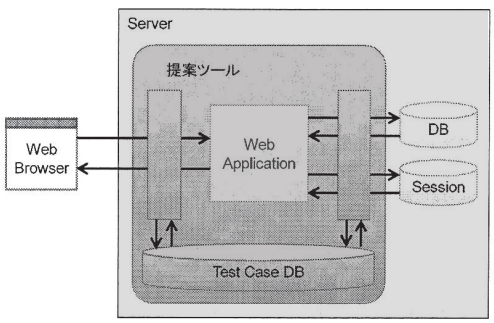

図 1　全体像

本ツールは，Web アプリケーションへの入力として，HTTP リクエスト (URL, GET/POST, Cookie) および，データベースとセッション変数の事前状態，そしてサーバの時刻を扱い，出力として，HTTP レスポンス (HTML, Header, Cookie) および，データベースとセッション変数の事後状態を扱う．

本ツールは，保存機構と再実行機構の 2 つからなる．保存機構は，回帰テストに先立ち，テストに使用した入出力を保存する機構である．再実行機構は，入力を復元し，出力を比較してテストを行う機構である．次に，それぞれについて述べる．

3.1 保存機構

保存機構では，まず，クライアントからアクセスを受けた直後に，クライアントからのリクエストおよび，サーバの状態を保存する．そして，クライアントにレスポンスを返す直前に，クライアントへのレスポンスおよび，サーバの状態を保存する．なお，本ツールでは，既存ツールのようにアサーションを作成する必要はない．

保存機構の動作を表したシーケンス図を，図 2 に示す．なお，opt は実行されてもされなくても良いことを表す．図 2 のように，ユーザは以下の手順でテストケースを作成する．

(1) ユーザは本ツールを介して，Web アプリケーションにアクセスする．本ツールは，ブラウザからのリクエストをアプリケーションに渡す前に，入力を保存する．また，レスポンスをブラウザに返す前に，出力を一時保存する．

(2) 本ツールは，ユーザに HTTP レスポンスおよびテストケースの ID を返す．得られた実行結果が正しいかどうか，ユーザが判断する．

(3) 実行結果が正しいと判断した場合には，一時保存した出力を本保存し，テストケースとする．以後，テストケースの ID を本ツールに渡すことで，回帰テストが可能となる．
実行結果が正しくない場合には，出力の保存は行わないが，テストケースの ID を本ツールに渡すことで，入力を再現し，再実行することが可能である．

3.2 再実行機構

再実行機構では，テスト対象アプリケーションが実行される前に，クライアントからのリクエストおよび，サーバの事前状態を復元する．そして，テスト対象の実行後，クライアントへのレスポンスおよびサーバの事後状態を，保存したものと比較する．なお，以前の出力が保存されていない場合でも再実行は可能であり，

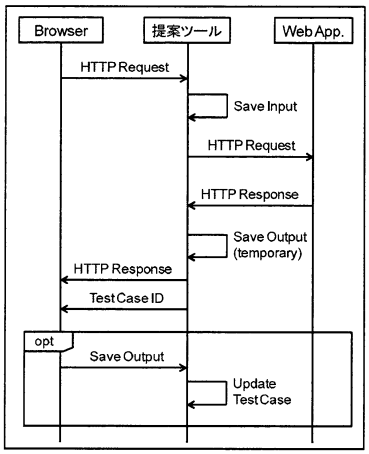

図 2 テスト保存の流れ

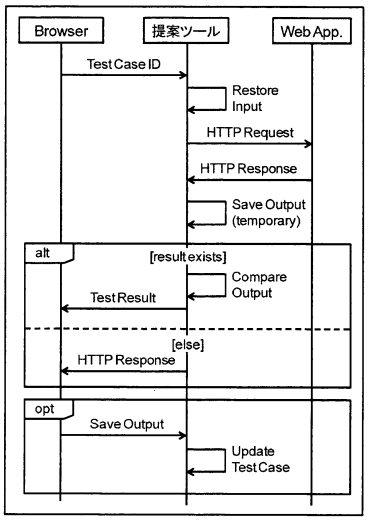

図 3 再実行の流れ

この場合，入力の復元だけを行い，比較は行わない．これは，デバッグ時等で，同じ入力を繰り返したい場合などに有用である．

再実行機構の動作を表したシーケンス図を，図 2 に示す．なお，alt は破線で仕切られた上下のどちらか一つが実行されることを表す．図 2 のように，ユーザは以下の手順で下記テストを行う．

(1) ユーザは本ツールに対し，テストケースの ID を渡す．本ツールは，テストケースの入力を復元し，Web アプリケーションを実行する．出力は，保存機構と同様，ブラウザに返す前に一時保存する．

(2) 以前の出力が存在する場合には，本ツールは今回の出力と比較し，比較結果をユーザに返す．以前の出力が存在しない場合には，保存機構と同様，ユーザにレスポンスを返す．

(3) 今回の実行結果を正しい出力として保存したい場合には，テストケースの上書きを行う．

なお，これは 1 回の回帰テストの動作だが，本ツールでは複数の回帰テストを一括で行うことができる．このため，保存されている全てのテストケースを一括で再テストすることにより，回帰テストを自動的に行うことができる．

4. 実　　装

本ツールは PHP[3] で実装された Web アプリケーションであり，テスト対象アプリケーションと同じディレクトリに配置する．本ツールが対象とするアプリケーションも PHP であり，データベースは MySQL を対象とする．

クライアントは，本ツールを通して，テスト対象アプリケーションにアクセスする．例えば，`/index.php` をテストする場合には，クライアントは次の URL にアクセスする．

`/phpRegTest.php/?testTarget=/index.php`

ここにアクセスすることで，本ツールは `/index.php` を内部で実行し，その結果をクライアントに返す．なお，再実行を行う際には，次の URL にアクセスする．

`/phpRegTest.php/?testId=<テストケース ID>`

以後，前者の，テストケースを作成するために最初に行う実行を，初回実行と呼び，後者の，実行を再現するための実行を再実行と呼び，区別する．

本ツールは，レスポンス HTML に加工を行う．まず，HTML に含まれるリンクを書き換え，本ツールを続けて使用できるようにする．例えば，`<a href="/a.php">` というリンクがあった場合，次のように書き換える．

`<a href="/phpRegTest.php?testTarget=/a.php">`

これにより，本ツールを使用していても，使用しない場合と同様のナビゲーションが可能となる．そして，HTML の末尾に，テストケースの ID や，結果を保存するためのボタン，実行結果の比較レポートなどを付与する．これらの加工により，本ツールを便利に使用できるようにしている．

また，サーバと Web アプリケーションの間のフックを実現するため，本ツールは，テスト対象アプリケーションを内部で実行する前に，プログラムを変換し，PHP の組み込み関数をフックするコードを挿入する．

本ツールは，ソースコードを読み込み，パースし，関数呼び出しを探す．そして，フックする関数の呼び出しを発見した場合，呼び出している関数名を置き換える．例えば，`func_call(...)` という関数呼び出しを，`__prt_hook_func_call(...)` のように置き換える．これにより，組み込み関数の呼び出しを，本ツールの定義する関数の呼び出しに置き換えることができる．なお，変換後のプログラムの実行は，PHP を評価して実行する組み込み関数である，`eval` を使用する．

また，`include` や `require` のような，外部の PHP を読み込んで評価する関数に対応するため，これらの関数もフックし，読み込む前にプログラム変換を行うコードを挿入する．

次に，HTTP リクエスト/レスポンス，セッション，データベース，サーバ時刻のそれぞれについて，入力の保存と復元，出力の保存と比較の実装を述べる．また，本ツールのテストケース管理機能についても述べる．

4.1 HTTP リクエスト/レスポンス

HTTP リクエストの保存は，テスト対象プログラムの実行前に行う．本ツールは，リクエストの情報が格納されている PHP スーパーグローバル変数 (`$_GET`, `$_POST`, `$_COOKIE` など) の値をシリアライズし，保存する．また，リクエストの復元は，テスト対象プログラムの実行前に，上記の変数の内容を保存した値で置き換えることにより実現している．

レスポンスの保存と比較は，HTTP Header, Cookie, 出力 HTML のそれぞれについて行う．Header と Cookie は，ヘッダの出力を行う組み込み関数 `header` , Cookie の出力を行う組み込み関数 `setcookie` をそれぞれフックし，出力される文字列を保存する．比較に関しては，単純に文字列比較を行う．

出力 HTML に関しては，単純に文字列比較を行うと，プログラムの動作に関係のない，ページデザインの変更があった場合にも，比較結果がエラーとなってしまう．このため，本ツールでは，全比較に加え，PHP の HTML 部 (`"<?" "?>"` で囲まれていない部分) を取り除いた比較を行う．これは，HTML 部では変数の値を出力することができないため，制御依存を除けば，プログラムの動作の影響を受けないためである．

まず，全比較は，テスト対象プログラムの実行前後に，組み込み関数である `ob_start` および `ob_end_flush` を使用することにより，対象プログラムの出力をバッファリングし，取得している．また，HTML 部を取り除いた比較は，PHP の出力関数をフックすることにより実現している．例えば，`echo`, `print` や，`<?=$val?>` をフックし，これらの出力を別途保存することにより，HTML 部以外の出力を取得している．

4.2 セッション変数

セッション変数に関する入出力の保存は，単純にテスト対象プログラムの実行前後に行う．具体的には，セッション変数が格納されている，`$_SESSION` をシリアライズし，保存する．復元は，`$_SESSION` の内容を，保存した値で置き換えることにより，実現している．また，結果の比較では，全てのキーと値のペアが完全に一致するかどうかを検査する．

4.3 データベース

プログラム変換を行い，PHP 組み込みの MySQL 関数をフックすることにより実現する．データベースの内容を全保存すると，内容によっては非常にコストがかかるため，SQL を解析し，必要最小限のテーブルのみを保存することにより，コストを抑えている．なお，テーブルの構造の保存・復元は行わず，現在の状態でテストを行う．これは，テーブルの構造の変化による不具合をテストできるようにするためである．

4.3.1 入力の保存

入力の保存では，クエリを送信する，`mysql_query` 関数をフックする．本ツールは，`mysql_query` に渡された SQL を解析し，アクセスするテーブルの内容を保存した後，`mysql_query` を実行する．例えば，“`SELECT * FROM book`” というクエリでは，`book` テーブルのみを保存する．なお，`mysql_query` が複数回呼ばれた場合，一度保存されたテーブルは保存しない．

これらの手順により，アクセスがあったテーブルの事前状態を，全て保存することができる．

4.3.2 入力の復元

入力の復元では，まず，テストケース生成時のテーブルの状態を再現したものを，本ツールの使用するデータベース内に作成する．そして，テスト対象プログラムのデータベースアクセスを，そこにリダイレクトさせることにより実現している．

まず，テスト対象プログラムの現在のテーブル構造をロードし，本ツールのデータベース内に作成する．この際，テーブル名は，“_<通し番号>_<元のテーブル名>_” のようにする．通し番号を付与するのは，同時に複数のテストを実行できるようにするためである．その後，作成したテーブルに，保存したデータをロードする．

次に，MySQL サーバに接続する `mysql_connect` 関数および，データベースに接続する `mysql_select_db`

関数をフックし，テスト対象プログラムが本ツールのデータベースへ接続するようにする．そして，mysql_query 関数をフックすることで，SQL に含まれるテーブル名を書き換え，データをロードしたテーブル名と一致させる．

これらの手順により，データベースからの入力を復元することができる．なお，本ツールのデータベースは，テスト対象のデータベースと一緒でも問題ない．このため，MySQL のアクセス権が低く，データベースを一つしか使用できないような環境でも，本ツールは使用可能である (テーブルを作成できる権限があればよい)．

4.3.3 出力の保存と比較

入力の保存時と同様，mysql_query 関数をフックし，SQL を解析し，アクセスするテーブルを解析する．出力の保存では，書き込みを行ったテーブルのみを保存すればよいため，SQL の SELECT 文に関しては，テーブルのリストから除外する．

また，出力では，入力と異なり，その場で保存するのではなく，アクセスしたテーブルのリストを作成し，テスト対象プログラムの実行終了後に，一括で保存する．すなわち，同じテーブルに複数回の書き込みが行われた場合，最終状態のみが保存される．

これらの手順により，書き込みがあったテーブルの事後状態を，全て保存することができる．また，データベースの出力の比較ではテーブル単位で内容の全比較を行う．

4.4 サーバ時刻

サーバ時刻に関しては，出力ではないため，入力の保存と復元のみ行う．保存は，テスト対象プログラムの実行前に一度だけ行い，復元では，time 等の日付関数をフックし，保存した値を返すようにする．日付関数が複数回呼ばれる場合にも，全て実行前に保存した値を返す．このため，最初に実行した結果と，再実行した結果は厳密には一致しない，しかし，再実行した結果と再々実行した結果は一致する．

4.5 テストケース管理機構

本ツールはテストケースを管理するためのインタフェースを持つ．本ツールのテストケース管理画面のスクリーンショットを図 4 に示す．

ユーザは，テストケース管理機構を用いて，個々のテストケースの入出力や，最新のテスト結果，最終テスト日時などを確認することができる．テスト結果は，HTTP リクエスト，データベースなどの各比較項目ごとにそれぞれ表示される．比較結果の詳細に関しては，リンクより確認することが可能である．また，管理画面では，個々のテストを個別に実行する機能に加え，複数のテストを一括して実行する機能を持つ．なお，一括で実行した場合，個々のレスポンスはユーザに返さず，本ツールは比較結果一覧の更新のみを行う．

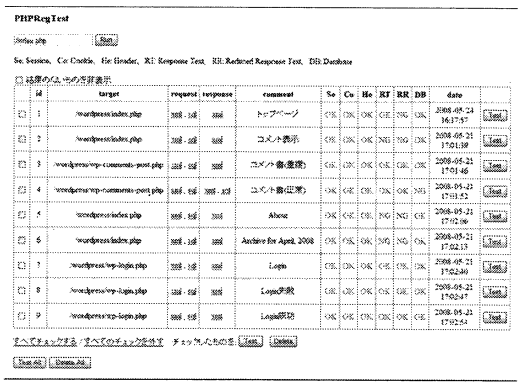

図 4 テストケース管理画面

5. 評　　価

本節では，本研究で提案するツールの適用例を示し，考察を行う．本ツールの適用例として，PHP で書かれたオープンソースのブログシステムである，WordPress[6] を対象とした．

まず，WordPress の様々なページを対象としたテストケースを作成し，本ツールがどれだけ実行を再現できるかどうかの実験を行った．そして，WordPress に対しいくつかの変更を加え，実際に回帰テストを行った．また，本ツールを介して対象アプリケーションを実行する際のオーバーヘッドを測定した．

評価環境は以下の通りである．

- CPU: Intel(R) Pentium(R) 4 CPU 3.00GHz
- メモリ: 1GB
- OS: Debian GNU/Linux 4.0
- Web サーバ: Apache HTTP Server 2.2.3
- PHP: PHP Version 5.2.0
- DB サーバ: MySQL 5.0.32
- WordPress: WordPress-2.5.1

5.1 再現能力の確認

テストケースを約 100 個作成し，本ツールが以前の実行を再現することができるかどうかを確認した．再実行同士，初回実行と再実行のそれぞれについて，対象プログラムの出力が一致しているかどうかを，本ツールの比較機能を用いて確認した．

5.1.1 再実行同士の比較

再実行同士では，次の 2 点を除いて，結果を再現することができた．

- ランダム文字列の出力
  e-mail を使用して記事を投稿するためのランダムキー生成などが再現できなかった．
- サーバ外への依存
  最新プラグインや，最も人気の高いプラグインを表示する機能が再現できなかった．これらの機能は外部のサーバから情報を取得していた．

5.1.2 初回実行と再実行の比較

初回実行と再実行の比較では，再実行同士で発生した相違に加え，時間に関する処理で相違が発生した．例えば，投稿日時の表示や，ページ生成にかかった時間の表示が再現できなかった．これは，第 4.4 節で述べたとおり，再実行では常にテスト対象プログラムの実行前に保存した値を返すためである．

5.1.3 考　　察

再現に関しては，一部再現できないものがあったが，ほぼ成功しており，特に，HTTP リクエスト，セッション，データベースからの入力に関しては問題なく再現できている．このため，デバッグ時等に現在の入力を再現したい場合には，本ツールは非常に有用であると言える．

ランダム文字列生成や時間に関しては，該当する関数が呼び出されるたびに値を記憶し，再現時には保存した値を順番に返すことにより，改善が可能であると考えられる．ただし，この場合でも，途中に時間関数が追加された場合には，順番が狂ってしまうため，完全に再現することはできない．

また，今回の評価では問題にならなかったが，本ツールはファイルの入出力を扱っていないため，ファイルについても考える必要がある．外部サーバへの依存も，ファイルの読み込みと考えることができるため，ファイルを扱うことにより，解決できると考えられる．実装方法としては，ファイル関数をフックして，アクセス先を本ツールが用意した別のファイルへの入出力にリダイレクトする方法が考えられる．

5.2 回帰テスト

WordPress に対し，実際に変更を加え，本ツールによる回帰テストのバグ発見能力を評価した．ここでは，例として，記事に対して言語情報を付与した際のシナリオを示す．具体的には，大きく分けて以下の二つの機能を追加した．

- 投稿時に言語を選ぶ機能
- 表示する記事を言語でフィルタする機能

まず，投稿データを格納している `wp_posts` テーブルに `language` フィールドを追加した．この時点で一度全体をテストし，問題が発生していないかどうかを確認した．特にバグは発見できなかったが，`wp_posts` テーブルへの出力があるものは，実際には問題ないにも関わらず，データベースの比較がエラーとなってしまった．これは，フィールドが増えているため，テーブルの比較で相違が生じるためである．そのため，この時点で一度テスト結果を上書きした．

次に，投稿ページに言語設定用のフォームを追加し，`language` フィールドに実際に書き込む処理を追加した．この時点で再び全体の回帰テストを行ったが，特にバグは発見できなかった．投稿機能は当然エラーとなるので，テストケースの上書きを行った．なお，投稿ページ自体は，フォームを追加しただけであり，PHP コードの HTML 部の変更にとどまっているので本ツールの比較機能では，エラーとならない．

最後に，閲覧ページに言語切り替えフォームを追加し，言語によるフィルタ機能を追加した．全体の回帰テストを行った結果，次のページでエラーが発生した．

- トップページ等の閲覧ページ
- コメント表示ページ
- 検索結果ページ
- RSS
- 投稿一覧 (管理ページ)

このように，想定していた閲覧ページだけでなく，さまざまなページに影響したことを検出することができた．この結果から，RSS ではフォームの他に別途フィルタ用の GET パラメータを用意する必要があることや，管理ページのような，フィルタの必要ない部分にもフィルタが有効になってしまったことがわかった．

5.2.1 考　　察

本ツールの回帰テスト機能により，プログラムの変更による，意図しない影響を自動的に検出することができた．本ツールを使用することで，複数の回帰テストを一括で自動的に実行することができるため，プログラムに変更を加えるたびに本ツールを使用することにより，プログラムの信頼性を確保することができると考えられる．

しかしながら，本ツールではテストが困難なケースも存在した．まず，Ajax ベースのロジックでは，ページ遷移が発生しないため，本ツールではテストケースを作成することができなかった．これには，ユーザ追加処理が該当した．そして，ページのリダイレクトを含むようなロジックはテストが困難であった．本ツールでは，ロジックの DB などへの出力と，結果ページへのリダイレクトがヘッダにあることは確認可能だが，リダイレクト先のページを確認することはできない．これには，ログイン処理が該当した．この問題に

対応するには，テスト後の状態を引き継いで次のテストを行う機能が必要であると考えられる．

また，データベースの構造を変更した場合にも，入力の再現ができないことがあった．例えばテーブルやフィールドを削除した場合や，名前を変更した場合には，テーブルの内容を復元するための SQL がエラーとなってしまい，入力を再現することができなかった．また，テスト時には存在しなかった新しいテーブルを追加した場合，そのテーブルは再現時には空となってしまう．このため，本来は正常に動作しているにも関わらず，エラーとなってしまうことがあった．例えば，コード内に埋め込んでいたマジックナンバー等を新しいテーブルから取得するようにした場合などが該当した．このように，データベースの正規化などにより，構造が大きく変更された場合には，本ツールを適用することができないという問題がある．対策として，無いテーブルやフィールドに関しては現在のデータを使用する等の方法が考えられる

5.3 実行時間

作成したテストケースを使用して，本ツールを使用しない実行・初回実行・再実行での実行時間の比を測定した．なお，測定は，ブログ記事を数件登録した状況で行った．測定した結果，ページによるばらつきは少なく，通常実行に比べ，初回実行では約 15 倍，再実行では約 40 倍のオーバーヘッドが生じた．

また，ブログデータを約 10 万件書き込み，トップページの実行時間を測定した．結果は以下の通りである．

- 本ツールを使用しない実行: 約 0.2 秒
- 初回実行: 約 14 秒 (約 70 倍)
- 再実行: 約 500 秒 (約 2500 倍)

このように，本ツールのオーバーヘッドは無視できないが，Web アプリケーションの特性上，レスポンスは 1 秒に満たないことがほとんどであるため，データベースの内容が少ない場合には，許容範囲であると考えられる．そして，開発段階で大量のデータがあることは少ないと考えられる．

オーバーヘッドの原因としては，まず，PHP のプログラム変換にかかる時間があげられる．WordPress は PHP では比較的大規模といえるアプリケーションであり，トップページでは，約 30,000 行の PHP を実行している．本ツールは，この 30,000 行を実行時に解析し，フックを挿入しているため，オーバーヘッドが発生している．

また，データベースの内容を多くした際に，再実行時間が跳ね上がっていることから，データベースの復元に非常に時間がかかっていることがわかる．そのため，保存・復元すべきデータをさらに減らす工夫が必要であると言える．本ツールでは，テーブル単位で保存を行っているが，SQL の WHERE 文も解析することにより，必要最小限のデータだけを保存する必要があると考えられる．

6. まとめ

本論文では，サーバサイドの入出力を考慮した，Web アプリケーションの回帰テスト支援ツールを提案した．本ツールを使用することで，テストに使用したクライアントとサーバの両サイドの入出力を保存することができる．本ツールは，保存した入力を再現することにより，一度行ったテストを再実行することができ，また，再実行した際の出力と，保存した出力を自動的に比較することができる．これらの機能により，Web アプリケーションの回帰テストを自動化することができる．

今後の課題としては，考察で述べた，ファイルへの対応，データベース構造の変化への対応，オーバーヘッドの削減などが挙げられる．

参考文献

1) Mehdi Jazayeri: Some Trends in Web Application Development, International Conference on Software Engineering 2007 Future of Software Engineering, pp.199-213, 2007.
2) The Apache Jakarta Projects: JMeter - Apache JMeter, *http://jakarta.apache.org/jmeter/*.
3) PHP: Hypertext Preprocessor, *http://www.php.net/*.
4) OpenQA: OpenQA: Selenium, *http://www.openqa.org/selenium/*.
5) Kristen R. Walcott, Mary Lou Soffa, Gregory M. Kapfhammer, and Robert S. Roos: Time-Aware Test Suite Prioritization, Proceedings of the 2006 International Symposium on Software Testing and Analysis, pp. 1-12, 2006.
6) WordPress, *http://wordpress.com/*.
7) Lei Xu, Baowen Xu, Zhenqiang Chen, Jixiang Jiang, and Huowang Chen: Regression Testing for Web Applications Based on Slicing, Proceedings of the 27th Annual International Computer Software and Applications Conference, pp. 652-656, 2003.