情報処理学会論文誌 Vol. 50 No. 12 3041–3051 (Dec. 2009)

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

今関雄人[†1] 高田眞吾[†1]

Webアプリケーションは機能の追加や変更が非常に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，クライアントからの入力を保存し，それを再現してテストを再実行するものがある．さらに，これらのツールは再実行時の実行結果を検査することも可能である．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にのみ着目しており，サーバサイドの入出力を扱うことができない．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．本ツールは入出力としてHTTPリクエスト/レスポンス，データベース，セッション変数，サーバ時刻を扱う．本ツールはこれらの入力を保存し，それを再現してテストを再実行することができる．さらに，本ツールはこれらの出力を保存し，再実行時の出力と比較することにより，テスト結果を自動的に検査することができる．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的に行うことができる．

Regression Testing Support Tool for Web Application Focusing on Server Side Input and Output

Yuto Imazeki[†1] and Shingo Takada[†1]

The regression testing of Web applications is very important because, compared with general applications, their requirements change more frequently. Some regression test support tools can save the input and re-execute previous tests by restoring saved input. Furthermore, these tools can automatically verify the re-executing output using assertion. However, these tools focus only on client side information and do not handle server side information. In this paper, we propose a regression test tool for Web applications, which supports both server side and client side information. For input and output, our tool handles HTTP request/response, database, session variable, and server time. Our tool can save and restore these inputs, save the resulting outputs and compare previous outputs with re-executing outputs for verification. These functions make possible the automatic regression testing of Web applications.

1. はじめに

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など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カ月，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[7]．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いえる．

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[11),20)]や，テストの実行順序に優先順位をつけるもの[12),16)]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．これらの研究は，すで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．しかしながら，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．このため，テストケースを用意する方法が必要である．

回帰テストでは，しばしば，以前に作成したテストケースを再実行し，プログラムが期待どおりに動作するかどうかを確かめる方法が用いられる[18]．すなわち，開発者は単体テストや統合テストで使用したテストケースを再利用し，回帰テストを行う．テストケースを効率的に再利用する方法として，capture/replayがある[10),13)]．この手法では，まず，ユーザからの入力といったテストの入力を保存（capture）し，テストケースを作成する．そして，保存した入力を再現することによりテストを再実行（replay）し，回帰テストを行う．capture/replayでは，実際にユーザが行った入力をそのままテストケースにすることができる．このため，テストケースの作成が非常に容易であり，また，実際の運用に近いデータをテストに使用することができるという特徴がある．

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，クライアント（Webブラ

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

ウザ）からの入力の capture/replay を行うものがある．これらのツールを使用することで，クライアントからの入力を保存し，それを再現してテストを再実行することが可能である．さらに，これらのツールは再実行時の実行結果を検査することも可能である．これらの機能を使用することで，テストを再利用し，回帰テストを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 HTTP 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．本ツールは，これらの入力を保存し，それを再現してテストを再実行することができる．さらに，本ツールはこれらの出力を保存し，再実行時の出力と比較することにより，テストの結果を自動的に検査することができる．本ツールを使用することで 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．

本論文の構成は次のとおりである．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 章ではサーバサイドの入出力を考慮した回帰テスト支援ツールを提案し，4 章ではツールの中心であるインタセプタの部分に注目して実装を述べる．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

本章では 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 既存ツール

Web アプリケーションの回帰テストを支援する既存ツールとして，Selenium [9]，JMeter [14]，WebVCR [1]，LogiTest [8]，MaxQ [15]，Badboy [2] などがある．

これらのツールは，いずれもクライアント側に配置するツールであり，サーバへのリクエスト（入力）を保存し，再実行を行うことができる．リクエストの保存方法はさまざまである．JMeter や MaxQ はプロキシサーバ機能を持っており，ブラウザでそのプロキシサーバを経由してアクセスすることにより，HTTP リクエストを保存することができる．Badboy や LogiTest は内部にブラウザを持ち，内部のブラウザを使用してテスト対象にアクセスすることで，HTTP リクエストを保存することができる．これらのツールは，保存した HTTP リクエストを再送信することにより，テストを再実行することができる．また，Selenium や WebVCR は Web ブラウザに表示されたページに対するユーザの操作（フィールドへの入力や，ボタンのクリックなど）を保存することができる．そして，ブラウザを自動的に操作することにより，入力を再現する．

さらに，Selenium や JMeter では，「アサーション」を用いて，実行結果を自動的に検査することができる．アサーションとは，実行結果（出力 HTML）の検査を行う命令であり，たとえば，ページタイトルが指定した文字列と一致しているかどうか，などを検査する．ただし，アサーションは，開発者が手動で作成する必要がある．

これらの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，実行結果を自動的に検査することができる．これらの機能により，1 度使用したテストを再利用し，回帰テストを自動的に行うことができる．

2.2 既存ツールの問題点

既存ツールは，いずれもクライアント側に配置するツールである．そのため，HTTP リクエストや HTTP レスポンスのような，クライアントサイドの入出力しか扱うことができない．しかしながら，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，データベースやセッション変数の内容により，動作が変化する．たとえば，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，商品の追加によりデータベースの内容が変わると，ページの内容も変化する．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，テストを作成したときと再テストするときでデータベースの内容が異なっていると，出力の HTML が変化してしまい，実行結果を正しく検査できないという問題がある．また，データベースなどへ書き込みを行うテストでは，サーバの状態が変わるため，再テストが困難な場合がある．たとえば，ユーザ登録機能では 1 度登録したユーザ名では登録できない．このため，最初のテストで登録に成功した場合，同じ入力で再テストを行うと，登録に失敗してしまう．

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．たとえば，商品の注文情報を書き込み，注文完了画面を表示するページでは，ページの内容だけでなく，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検査する必要がある．しかしながら，既存ツールで

は，HTTP レスポンスしか検査することができないという問題がある．このように，既存ツールはクライアントサイドの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．

サーバサイド，特にデータベースのテストに関する研究としては，テスト入力を自動生成するもの[3),5)]，回帰テスト選択手法[4),17)]，そして，回帰テストの実行順序に関するもの[6)]がある．しかしながら，データベースやセッション変数の現在の状態を保存，再現するような研究は存在しない．

3. サーバサイドの入出力を考慮した回帰テスト支援ツール

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，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），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，テスト時の出力を保存し，再実行時の出力と比較することにより，テスト結果を自動的に検査する機能を持つ．このため，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．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．

本ツールは PHP で記述された Web アプリケーションを対象とし，本ツール自身も PHP で記述された Web アプリケーションである．本ツールはテスト対象アプリケーションと同じサーバに配置するツールであり，テスト対象の動作するサーバ内に，仮想ホスト（仮想的な別の Web サーバ）を作成し，その中に配置する．この仮想ホストにアクセスすることにより，本ツールはクライアントと Web アプリケーション間に加え，サーバ（データベースやセッション）と Web アプリケーション間のメッセージをインタセプトする．これにより，クライアント側だけでなく，サーバサイドの入出力も扱うことができる．サーバと Web アプリケーション間をインタセプトするために，本ツールはテスト対象プログラムを実行時に変換し，本ツールの内部で実行する（詳細は第 4.1 節で述べる）．

本ツールは，テストを保存するための保存機構と，再実行するための再実行機構，および，テストケース管理機構からなる．次に，それぞれについて述べる．

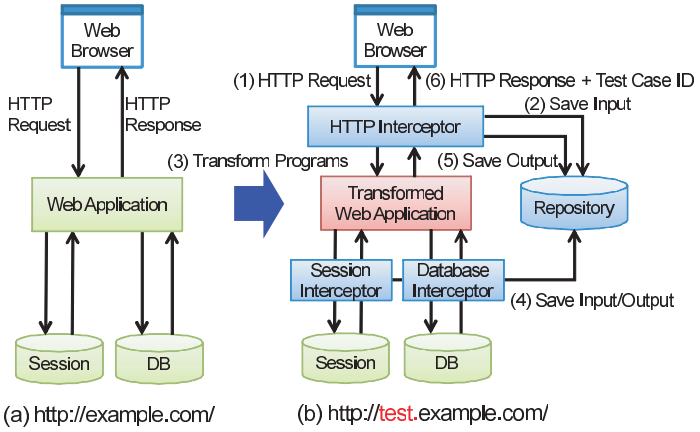

図 1　保存機構
Fig. 1　Saving input/output.

3.1 保 存 機 構

保存機構は，回帰テストに先立ち，テストの入出力を保存する機構である．

図 1 (a) は，データベースやセッション変数を使用する Web アプリケーションのアーキテクチャである．Web アプリケーションは HTTP リクエストを受け取り，セッション変数やデータベースにアクセスを行い，HTTP レスポンスを返す．本ツールはこの 1 回の HTTP リクエストから HTTP レスポンスまでを 1 つのテストケースとして扱う．

本ツールは，図 1 (b) のように，クライアントと Web アプリケーションの間および，サーバと Web アプリケーションの間にインタセプタを配置する．本ツールは次の手順で入出力の保存を行う．

(1)　ユーザは，Web ブラウザを利用し，本ツールを介してテスト対象 Web アプリケーションにアクセスする．たとえば，`http://example.com/` をテストするには，仮想ホスト `test.example.com` を作成し，`http://test.example.com/` にアクセスする．このように，リクエストするドメインを，本ツールを配置したドメインに切り替えるだけで，本ツールを使用することが可能である．

(2)　HTTP インタセプタは，クライアントからの入力（HTTP リクエスト）を保存する．

(3)　本ツールはテスト対象プログラムを変換し，本ツールの内部で実行する

(4)　テスト対象はインタセプタを経由してセッションやデータベースにアクセスする．イ

ンタセプタは，その入出力を保存する．

(5)　テスト対象の実行後，HTTP インタセプタはクライアントへの出力（HTTP レスポンス）を保存する．

(6)　本ツールは，HTTP レスポンスおよびテストケース ID をクライアントに返す．ユーザは，今回保存した入出力をテストケースとして登録するか，破棄するかを選ぶ．

本ツールは，レスポンスをクライアントに返す前に，HTML に加工を行う．まず，HTML の末尾に，テストケースの ID と，保存，再実行ボタンを付与する．保存を押すことにより，今回の入出力を保存し，テストケースとすることができる．再実行ボタンを押すことで，今回の入力を再現し，再実行することができる．また，HTML に含まれるリンクを書き換え，そのページからのリンクも，本ツールを介してアクセスするように変更する．相対パスはそのまま使用可能であるため，絶対パスで指定されている内部リンクのみ書き換えを行う．

3.2　再実行機構

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．再実行の際，インタセプタは保存した入力を復元し，テスト対象に送信することで，テスト対象への入力を代行する．また，インタセプタはテスト対象からの出力を保存し，今回の出力と以前の出力との比較を行うことで，テスト結果を検査する．

本ツールは次の手順でテストの再実行を行う（**図 2**）．なお，次の手順は 1 回のテスト再実行の手順だが，本ツールでは複数のテストを一括で実行することもできる．

(1)　ユーザは本ツールにテストケース ID を渡す．具体的には，次のように，URL の末尾にテストケース ID を付与してアクセスする．

```
http://test.example.com/?testId=1
```

(2)　各インタセプタはテストケースリポジトリより保存した入力を読み込む．

(3)　本ツールはテスト対象プログラムを変換し，復元した HTTP リクエストを使用して実行する．

(4)　テスト対象の実行中，インタセプタはテスト対象への入力を代行する．また，出力を保存し，以前の出力と比較することで，テスト結果を検査する．

(5)　テスト対象の実行終了後，HTTP インタセプタは，クライアントへの出力を保存し，以前の出力と比較することで，テスト結果を検査する．

(6)　本ツールは，HTTP レスポンスおよびテスト結果（出力の比較結果）をクライアントに返す．ユーザは，必要に応じ，今回の出力で以前の出力を上書きすることができる．

本ツールは，保存時と同様，再実行時にもレスポンス HTML の加工を行う．本ツールは

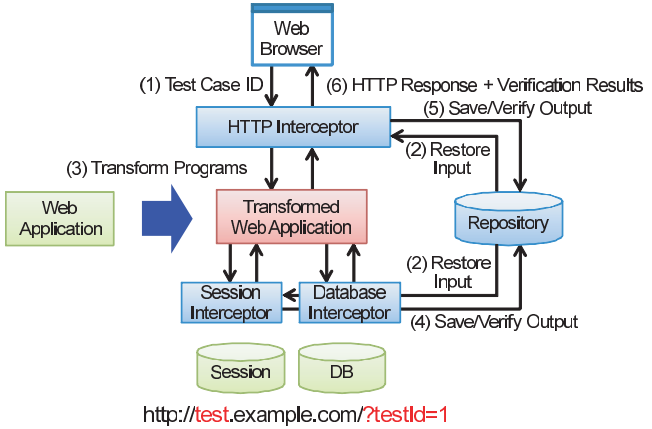

図 2　再実行機構
Fig. 2　Reexecuting a test case.

HTML の末尾に，テスト結果，上書きボタン，そして再実行ボタンを付与する．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する．また，保存時と同様，リンクの書き換えを行う．

3.3　テストケース管理機構

本ツールのテストケース管理機構は，各テストの入出力に加え，テスト結果（出力の比較結果）を管理している．テストケース管理画面のスクリーンショットを**図 3** に示す．

テストケース管理画面では，テスト ID，テスト対象 URL，テスト入出力の詳細へのリンク，テストケースのコメント，最新のテスト結果，最終テスト日時テストを再実行するためのボタン，そして，テストを一括で再実行/削除するためのチェックボックスが，表形式で表示される．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．テスト結果の詳細はリンクより確認することが可能である．また，すべてのテストを一括で再実行するためのボタンを持つ．

なお，複数のテストを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，テスト結果の更新のみを行う．このため，一括テスト後，ユーザはテストケース管理画面でテスト結果の一覧を確認するだけでよい．

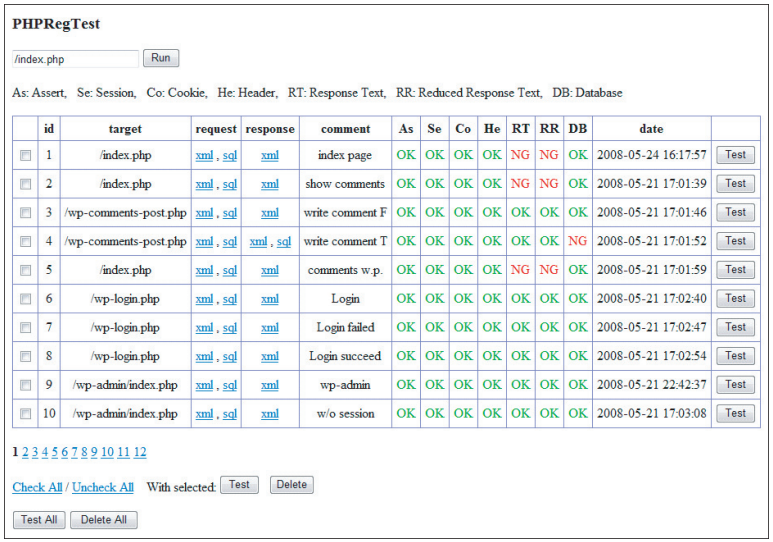

図 3　テストケース管理画面
Fig. 3　UI for test case management.

4. インタセプタ

本章では，3 章で述べたインタセプタの詳細について述べる．まず，インタセプタを実現するためのプログラム変換について述べ，その後，各インタセプタについて述べる．

4.1 プログラム変換

本ツールを配置した仮想ホストにアクセスがあった場合，まず，本ツールは Apache のモジュールである mod_rewrite を使用して URL の書き換えを行い，URL を本ツールへのパラメータとして取得する*1．

次に，本ツールはテスト対象の配置されたディレクトリから，URL に対応するファイルを取得し，クライアントに返す．アクセス先のファイルが PHP である場合，本ツールはテスト対象の PHP を本ツールの内部で実行し，その結果を出力する．この際，本ツールはテスト対象プログラムの変換を行い，Web アプリケーションとクライアント/サーバ間をインタセプトするためのコードを挿入する．

PHP プログラムでは，データベースやセッションへのアクセスに，PHP 組み込みの関数を用いる．そこで，本ツールはテスト対象プログラムから，特定の組み込み関数の呼び出しを探し，本ツールの定義した関数の呼び出しに置き換える．たとえば，`func(...)` という関数呼び出しを，`__prt_func(...)` のように置き換える．これにより，組み込み関数本来の処理を実行する代わりに，本ツールの処理を実行することができる．以後，本論文中では，このことを「`func` をフックする」と呼ぶ．変換後のプログラムの実行には，文字列を PHP プログラムとして実行する組み込み関数である，`eval` を使用する．また，PHP には，`include` や `require` のような，別ファイルの PHP を読み込んで実行する関数が存在する．これらの関数に対応するために，本ツールはこれらの関数をフックし，プログラムを読み込み，変換し，実行するという一連の動作を行う関数の呼び出しに置き換える．

このように，本ツールは，テスト対象プログラムを実行時に変換し，本ツール内で実行する．これにより，テスト対象の実行前後および，テスト対象の実行中に本ツールの処理を実行することができる．これにより，各種インタセプタを実現している．なお，本ツールは，プログラムをあらかじめ変換しておくのでなく，実行時に変換する．これは，あらかじめ変換する方法では，ユーザはテスト前に明示的に変換を行う必要があり，デバッグ時など，ソースを修正しながら同じ入力を繰り返したい場合に非常に煩雑となるためである．実行時に変換を行うことにより，本ツールを配置したドメインにアクセスするだけで，それ以外の操作をせずに，最新のソースコードの状態で本ツールを使用することができる．

次に，各インタセプタについて，入力の保存と復元，出力の保存と比較の詳細を述べる．

4.2 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．そして，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．比較では，単純に文字列比較を行う．

出力 HTML は，単純に文字列の全比較を行うと，プログラムの動作に関係のないページ

*1 テスト対象アプリケーションが mod_rewrite を使用している場合，本ツールの mod_rewrite の設定の前に，テスト対象の mod_rewrite の設定を付与する必要がある．

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 が動的に出力したテキストのみの比較を行う．PHP では，<?php と ?> で囲まれている部分以外は静的な HTML と同様に扱われるため，この部分を取り除くことにより，動的出力のみを抽出する．

まず，全比較は，PHP の出力をバッファリングすることにより実現している．すなわち，PHP がクライアントにレスポンスを返す前に，レスポンスの内容を本ツールにより取得し，保存および比較を行う．そして，動的出力のみの比較は，PHP の出力関数をフックすることにより実現している．echo，print や，<?= ?> のような出力を行う関数をフックし，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．

4.3　セッション変数

PHP では，組み込み関数である session_start を呼び出すことにより，セッションが開始され，スーパグローバル変数 $_SESSION に，セッション変数の値が格納される．

そこで，入力の保存と復元では，この関数のフックを行う．入力の保存は，session_start の実行後，$_SESSION の値を保存することにより実現している．入力の復元は，本来の session_start の処理を置き換え，保存した値を $_SESSION にロードすることにより実現している．

出力の保存と比較は，テスト対象の実行後に行う．すなわち，テスト対象の実行後，$_SESSION の内容を保存する．出力の比較は，$_SESSION が連想配列であるため，各キーが過不足なく存在しているかどうか，そして，各キーの値が一致しているかどうかを検査する．

4.4　データベース

データベースの保存と復元は，PHP 組み込みの MySQL 関数をフックすることにより実現している．本ツールは，テーブルのデータの保存と復元を行うが，テーブルの構造はそのまま復元せず，現在のテーブルの構造を用いる．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．また，すべてのテーブルのデータを保存すると，データによっては非常にコストがかかるため，SQL を解析することにより，必要最小限のテーブルのみを保存する．

4.4.1　入力の保存

入力の保存は，データベースにクエリが送信されるタイミングで行い，アクセスするテーブルのみ保存を行う．具体的には，データベースにクエリを送信する mysql_query 関数をフックし，次の処理に置き換える．

(1)　SQL クエリがアクセスするテーブルの抽出
　　MySQL の文法に従い，SQL クエリを解析する．たとえば，SELECT 文であれば FROM 句，INSERT 文であれば INTO 句を解析することにより，アクセスするテーブルを抽出することが可能である．

(2)　アクセスするテーブルの構造とデータを保存
　　なお，mysql_query が複数回呼ばれた場合，1 度保存したテーブルは保存しない．

(3)　mysql_query を実行

たとえば INSERT INTO book SET ... というクエリが来た場合，本ツールは book テーブルを保存する．その次のクエリが SELECT * FROM book, order であった場合，order テーブルのみを保存し，すでに保存した book テーブルは保存しない．最初のクエリと 2 番目のクエリで book テーブルのデータは変化するが，2 番目のクエリでの book のデータは，最初のクエリを再実行することで再現することが可能である．このため，テーブルの初期状態のみ保存すればよい．

4.4.2　入力の復元

入力の復元は，テストケース生成時を再現したデータベースを作成し，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することにより実現している．

テスト対象の実行前に，本ツールはまず，データベースサーバ内に新しいデータベースを作成し，保存したテーブルの構造とデータをロードする．これにより，テストケース生成時の各テーブルの構造とデータを再現することが可能である．次に，本ツールはこのデータベースに現在のテーブル構造を適用する．復元した各テーブルと，現在のテーブルを比較し，フィールドが追加されていれば，そのフィールドを追加し（フィールドの値は空またはデフォルト値となる），フィールドが削除されていれば，そのフィールドを削除する．

データベースアクセスのリダイレクトは，データベースサーバに接続する mysql_connect 関数および，使用するデータベースを選択する mysql_select_db 関数をフックすることにより実現している．本ツールは，これらの関数に渡される引数を書き換えることにより，テスト対象のデータベースアクセスを，本ツールが再現したデータベースへリダイレクトする．

4.4.3　出力の保存と比較

出力の保存は，入力の保存と同様，mysql_query 関数をフックして SQL クエリの解析を行う．本ツールは，出力としてデータベースの事後状態，すなわちアクセスされたテーブルの最終状態を扱う．このため，クエリ送信時には，テーブルの保存は行わず，アクセスされたテーブルのリストを作成するのみであり，テスト対象プログラムの実行終了後に一括で保

存を行う．したがって，同じテーブルに複数回の書き込みが行われた場合，最終状態のみを保存する．なお，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 文のみがアクセスするテーブルはリストに含めない．

また，出力の比較では，テーブル単位で各フィールドの内容をすべて比較する．この比較はデータベース上（入力の復元時に作成したデータベース）で SQL を使用して行う．まず，復元した各テーブルの名前を変更し，プレフィックスを付与する．そして，以前の出力をデータベース内にインポートし，対応するテーブル同士で比較を行う．なお，テーブルに新しいフィールドが追加されている場合や，逆に削除されている場合には，そのフィールドは比較対象には含めない．

4.5　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 1 度だけ行う．そして，復元では `time` などの日付/時刻関数をフックし，保存した値を返すようにする．本ツールは，日付/時刻関数が複数回呼ばれる場合にも，すべて実行前に保存した値を返す．このため，プログラムの実行中に経過する時間が無視されてしまい，厳密には保存時の実行を再現することができない．そこで，本ツールでは，保存時の時刻と再実行時の時刻を一致させるため，保存時にも日付/時刻関数をフックし，実行前に保存した値を返す．

5. 適 用 事 例

本章では，本研究で提案するツールの適用例を示し，考察を行う．PHP で記述されたオープンソースのブログシステムである WordPress [19] を対象とし，本ツールと既存ツール（Selenium，JMeter）を比較しながら実験を行った．

まず，WordPress のさまざまなページを対象としたテストケースを，本ツールと既存ツールを用いて，それぞれ作成した．そして，WordPress に何も変更を加えない状態で，テストの再実行を行い，テストの再現能力を比較した．次に，WordPress に対しいくつかの変更を加え，作成したテストケースを使用して，実際に回帰テストを行った．また，本ツールでテスト対象プログラムを実行する際に生じるオーバヘッドを測定した．

実験環境は以下のとおりである．

- CPU：Intel(R) Pentium(R) 4 CPU 3.00 GHz
- メモリ：1 GB
- 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

なお，Selenium と JMeter では実験結果にほぼ相違がなかったため，以下では Selenium，JMeter を区別せず，単に「既存ツール」と記す．

5.1　再 現 能 力

WordPress に何も変更を加えない状態で，テストの再実行を行い，本ツールと既存ツールのテスト再現能力を比較した．テスト対象に変更を加えていないため，出力の検査でエラーを検出した場合には，テストを再現できていないことになる．

5.1.1　記事一覧の表示

WordPress では，トップページなど，ブログ記事の一覧を表示するページが存在する．記事一覧のページでは，新着順に表示を行うため，新しい投稿を行うと表示する内容が変化する．このため，新しい投稿を行うたびに，既存ツールはバグが発生したと誤検出してしまった．しかしながら，本ツールは以前のデータベースの内容を再現できるため，新しい投稿を行っても表示する内容は変わらず，バグの誤検出は発生しなかった．

5.1.2　記事の投稿

WordPress でブログ記事を投稿すると，記事の投稿が完了した旨が画面に表示される．そのため，既存ツールではこの表示により投稿エラーが発生していないかどうかを検査した．しかしながら，この表示では投稿エラーが発生しなかったことは分かるが，実際にデータベースに書き込みができたのかどうかまでは確認することができない．一方，本ツールは投稿処理後のデータベースの内容を確認することができるため，実際に書き込みが行われたかどうかを検査することができた．

5.1.3　ユーザ登録

WordPress には，ブログを投稿するためのユーザを作成する，ユーザ登録機能がある．ユーザ登録の入力項目には，ユーザ名やメールアドレスなどのユニークな項目が存在し，1 度登録したユーザ名やメールアドレスでは登録することができない．このため，既存ツールでテストを再実行した際，ユーザ名がすでに存在するために登録エラーとなり，バグと判断してしまった．一方，本ツールはまだユーザ名が登録されていない状態のデータベースを再現したため，再実行を行っても登録エラーにはならず，バグと判断することはなかった．

5.1.4　ログインが必要なページ

WordPress には，記事の投稿や削除を行うための管理ページがあり，管理ページにアクセスするにはログインが必要である．ログインが必要なページをテストする際，既存ツールでは，まずログインし，その後テスト対象ページへアクセスするというテストケースを作成する必要があった．しかしながら，本ツールはサーバの状態を復元し，ログインしているという状態を復元することができる．このため，テスト対象ページにのみアクセスするテストケースを直接作成することができた．

なお，ログイン情報にはセッション変数を使用することが多いため，本ツールのセッション変数の保存/復元機能により復元することが可能である．しかし，WordPress はセッション変数を用いず，認証キーをクッキーに保存する仕組みを使用していた．この認証キーには有効期限が埋め込まれており，照合時に参照されていた．このため，既存ツールでログインが必要なページを再実行すると，有効期限内は動作するが，期限後は動作しなくなってしまった．一方，本ツールではサーバ時刻も再現するため，有効期限切れにならなかった．

5.1.5　考　　察

本ツールはサーバの状態を再現できるため，既存ツールでは再現できないテストも再現することができた．このため，回帰テスト時だけでなく，デバッグ時などに現在の入力を再現したい場合にも，本ツールは非常に有用であるといえる．

なお，既存ツールでサーバサイドの入出力を考慮したテストを行う場合，データベースやセッション変数の状態を設定するためのスクリプトを作成し，テスト前に実行するという手法が用いられる．これにより，サーバサイドの状態を再現することができるが，このようなスクリプトは手動で作成する必要がある．このため，既存ツール本来の用途である capture/replay による，テストケース作成の容易さが失われてしまう．一方，本ツールはサーバサイドの入出力に関しても capture/replay を行うことができるため，テストケースの作成が非常に容易である．

しかしながら，次の 2 点は，本ツールでも再現することができなかった．

- 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．

また，今回の適用例では問題なかったが，本ツールはファイルの入出力を扱うことができない．このため，投稿内容をファイルに書き込み，表示するようなアプリケーションは扱うことができない．

ファイルを扱う方法としては，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．なお，外部サーバへの依存も，ファイルの読み込みの一種と考えることができる．ランダム文字列に関しては，ランダム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返すことにより，再現可能であると考えられる．もしくは，srand などの乱数シードを作成する関数をフックする方法も考えられる．

5.2　回帰テスト

WordPress に対しいくつかの変更を加え，作成したテストケースを使用して，実際に回帰テストを行った．

5.2.1　投稿/記事一覧表示の拡張

投稿時に言語を選ぶ機能を追加し，記事一覧表示に言語フィルタ機能を追加した．この変更によりバグは発生しなかったが，開発中に投稿のテストを行ったため，既存ツールでは記事管理ページなどのさまざまなページでバグが発生したと誤検出してしまった．これは，5.1.1 項で指摘した問題と同様である．本ツールでは以前のデータベースの内容を再現できるため，バグの誤検出は発生しなかった．

5.2.2　記事投稿の仕様変更

記事の投稿機能に変更を加え,「コメントを許可」のチェック項目を削除し，つねにコメントを許可するように変更した．チェック項目の削除にともない，データベースから対応するフィールドを削除した．

この状態で既存ツールと本ツールで回帰テストを行ったところ，既存ツールではバグを検出できなかったが，本ツールでは記事の編集が失敗するというバグを検出することができた．すなわち，投稿機能への変更が，編集機能へ影響を与えたことを検出することができた．

このバグは，データベースのフィールドを削除したにもかかわらず，編集時の UPDATE クエリが削除したフィールドにアクセスしていたことが原因であり，WordPress は UPDATE クエリが実際には失敗しているにもかかわらず，記事の編集が完了したと表示していた．5.1.2 項で指摘したように，既存ツールはこの表示のみを確認しており，データベースの内容を検証していないため，バグを検出することができなかった．一方，本ツールはデータベースの内容も検証したため，バグを検出することができた．

5.2.3　ユーザ登録機能の拡張

ユーザ登録機能を拡張し，登録時の必須入力項目として「秘密の質問」を追加した．

WordPress のユーザ登録機能は 2 種類あり，通常の機能である,「管理者による登録」と，オプションで有効になる，一般的なコミュニティサイトのような,「ユーザ自身による登録」がある．本事例では,「管理者による登録」のみしか変更を行わなかったため,「ユーザ自身による登録」において，必須入力項目が空になってしまい，登録エラーが発生するというバグが発生した．

本ツールを使用することにより，このバグを検出することができた．なお，このバグは既存ツールでも検出できたが，5.1.3 項で指摘したように，既存ツールでは，ユーザ登録を再現できず，つねにバグが発生していると誤検出してしまった．このため，既存ツールでは本当にバグが発生したのかどうかが判断できなかった．しかし，本ツールでは機能の拡張を行った後にのみバグを検出したため，バグが発生したことが分かった．

5.2.4　考　　察

本ツールはサーバサイドの入力を再現し，出力を検査できるため，既存ツールでは検出できないバグを発見することができ，バグを誤検出することもなかった．

なお，既存ツールではアサーションを使用して出力の検査を行うが，入力の再現と同様に，スクリプトを使用することにより，サーバサイド出力の検査を行うことも可能である．しかしながら，既存ツールでは，アサーションやこのようなスクリプトは手動で作成する必要がある．一方，本ツールは，出力を保存し比較することにより，出力の検査を自動的に行うことができる．したがって，アサーションもサーバサイドの検査スクリプトも作成する必要はなく，テストケースの作成が非常に容易である．

このように，本ツールを使用することにより，単体テストや統合テストのテストケースを効率的に再利用し，回帰テストを行うことができる．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，データベースのテーブル構造を変更した場合，本ツールではテストが困難なケースがあった．たとえば，データベースにフィールドを追加した場合や，フィールド名を変更した場合，その変更はテストケースとして保存したデータベースには反映されない．このため，本ツールが復元したデータベースでは，そのフィールドは空となる．具体的には，記事のタイトルを保持しているフィールド名を変更した場合，タイトルを取得するクエリも正しく変更していれば，既存ツールで作成したテストケースは，そのまま使用可能である．しかしながら，本ツールでは，記事のタイトルが空となってしまうため，バグが発生したと誤検出してしまった．

この問題を解決するためには，な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方法や，データベース構造の変更履歴を保存して追跡する方法が考えられる．

表 1　トップページのオーバヘッド（単位：秒）
Table 1　Overhead in top page (Time in secs).

	10 articles			10,000 articles			100,000 articles		
	w/o	save	reex	w/o	save	reex	w/o	save	reex
Trans.	-	1.72	1.69	-	1.71	1.70	-	1.72	1.70
DB	-	0.18	0.18	-	0.46	1.68	-	3.03	13.65
Comp.	-	-	0.10	-	-	0.64	-	-	0.54
Other	0.22	0.24	0.24	0.27	0.41	0.89	0.27	0.42	5.59
Total	0.22	2.15	2.22	0.27	2.60	4.92	0.27	5.18	21.50

Trans.　プログラム変換に要した時間
DB　データベースの保存/復元に要した時間
Comp.　出力の比較に要した時間
Other　それ以外（テスト対象の実行，DB 以外の保存/復元など）

表 2　投稿処理のオーバヘッド（単位：秒）
Table 2　Overhead in posting a article (Time in secs).

	10 articles			10,000 articles			100,000 articles		
	w/o	save	reex	w/o	save	reex	w/o	save	reex
Trans.	-	2.11	2.30	-	2.24	2.31	-	2.33	2.13
DB	-	0.24	0.23	-	0.82	2.01	-	6.02	17.16
Comp.	-	-	0.10	-	-	4.09	-	-	49.86
Other	0.25	0.30	0.30	0.31	0.36	0.35	0.73	0.80	0.76
Total	0.25	2.66	2.93	0.31	3.43	8.78	0.73	9.16	69.92

5.3　オーバヘッド

本ツールのオーバヘッドを測定するため，WordPress のトップページ（約 30,000 LOC）および，記事の投稿処理（約 37,000 LOC）の実行時間を測定した測定結果を**表 1**，**表 2** に示す．

測定は，次の (1) 本ツールを使用しない場合（w/o），(2) テスト保存時（save），(3) 再実行時（reex）の 3 つの実行について行った．まず，WordPress の初期状態（記事が 1 件のみ登録されている状態：合計 198 レコード）で測定を行い，その後，記事を 1 万件，10 万件と変化させ，測定を行った．

なお，測定結果はサーバ上の実行時間であり，通信にかかった時間は含まれていない．また，既存ツールではサーバ側にオーバヘッドが生じることはない（クライアント側の処理にはオーバヘッドが生じるが，十分に小さい）．

表 1，表 2 から，データベースに含まれるデータが少ない場合は，プログラム変換がオーバヘッドの主要因となっていることが分かる．このオーバヘッドはプログラムを実行時に解析していることが原因であり，LOC に依存している．測定対象は約 30,000 LOC 以上という比較的大規模なページであるため，このオーバヘッドは実用的な範囲内であるといえる．

また，データベースに含まれるデータが増えるにつれ，データベースの処理時間が増大しており，特にデータベースの復元と比較に時間がかかっている．なお，トップページではデータベースへの書き込みがないため，出力の保存と比較がなく，オーバヘッドが少なくなっている．1 万件の状態では実用的な範囲内に収まっているといえるが，10 万件登録した状態では，本ツールのオーバヘッドは無視できないほど大きい．しかしながら，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ため，本ツールのオーバヘッドは実用的な範囲内であるといえる．

本ツールは，テーブルの内容をすべて保存しているために，データが多い場合のオーバヘッドが大きくなっている．このオーバヘッドを削減するために，SQL の WHERE 句も解析することにより，保存するデータを削減する方法が考えられる．

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，1 度行ったテストを再実行することができる．また，再実行した際の出力と，保存した出力を比較し，テストの結果を自動的に検査することができる．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，5.1.5 項で述べた，ファイルや乱数への対応や，5.2.4 項で述べたデータベース構造の変化への対応，そしてオーバヘッドの削減があげられる．

参 考 文 献

1) Anupam, V., Freire, J., Kumar, B. and Lieuwen, D.: Automating Web Navigation with the WebVCR, *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol.33, No.1-6, pp.503–517 (2000).
2) Badboy Software: Badboy. http://www.badboy.com.au/ (accessed 2009-03-19).
3) Chays, D., Deng, Y., Frankl, P.G., Dan, S., Vokolos, F.I. and Weyuker, E.J.: An AGENDA for Testing Relational Database Applications: Research Articles, *Journal of Software Testing, Verification and Reliability*, Vol.29, No.1, pp.17–44 (2004).
4) Daou, B., Haraty, R.A. and Mansour, N.: Regression Testing of Database Applications, *Proc. 2001 ACM Symposium on Applied Computing*, pp.285–289 (2001).
5) Deng, Y., Frankl, P. and Wang, J.: Testing Web Database Applications, *ACM SIGSOFT Software Engineering Notes*, Vol.29, No.5, pp.1–10 (2004).
6) Haftmann, F., Kossmann, D. and Kreutz, A.: Efficient Regression Tests for Database Applications, *Proc. 2005 Conference on Innovative Data Systems Research*, pp.95–106 (2005).
7) Jazayeri, M.: Some Trends in Web Application Development, *ICSE 2007, Future of Software Engineering*, pp.199–213 (2007).
8) LogiTest: LogiTest. http://logitest.sourceforge.net/ (accessed 2009-03-19).
9) OpenQA: Selenium. http://seleniumhq.org/ (accessed 2009-03-19).
10) Orso, A. and Kennedy, B.: Selective Capture and Replay of Program Executions, *Proc. 3rd International Workshop on Dynamic Analysis*, pp.1–7 (2005).
11) Rothermel, G. and Harrold, M.J.: A Safe, Efficient Regression Test Selection Technique, *ACM Trans. Software Engineering and Methodology*, Vol.6, No.2, pp.173–210 (1997).
12) Rothermel, G., Untch, R.J. and Chu, C.: Prioritizing Test Cases For Regression Testin, *IEEE Trans. Softw. Eng.*, Vol.27, No.10, pp.929–948 (2001).
13) Steven, J., Chandra, P., Fleck, B. and Podgurski, A.: jRapture: A Capture/Replay Tool for Observation-based Testing, *Proc. 2000 ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp.158–167 (2000).
14) The Apache Jakarta Projects: Apache JMeter. http://jakarta.apache.org/jmeter/ (accessed 2009-03-19).
15) Tigris.org: MaxQ. http://maxq.tigris.org/ (accessed 2009-03-19).
16) Walcott, K.R., Soffa, M.L., Kapfhammer, G.M. and Roos, R.S.: Time-Aware Test Suite Prioritization, *Proc. 2006 International Symposium on Software Testing and Analysis*, pp.1–12 (2006).
17) Willmor, D. and Embury, S.M.: A Safe Regression Test Selection Technique for Database-Driven Applications, *Proc. 21st International Conference on Software Maintenance*, pp.421–430 (2005).
18) Wong, W.E., Horgan, J.R., London, S. and Agrawal, H.: A Study of Effective Regression Testing in Practice, *Proc. 8th International Symposium on Software Reliability Engin.*, pp.264–274 (1997).

19) WordPress.com: WordPress. http://wordpress.com/ (accessed 2009-03-19).
20) Xu, L., Xu, B., Chen, Z., Jiang, J. and Chen, H.: Regression Testing for Web Applications Based on Slicing, *Proc. 27th Annual International Conference on Computer Software and Applications*, pp.652–656 (2003).

(平成 21 年 3 月 25 日受付)
(平成 21 年 7 月 2 日採録)

今関　雄人

2005 年慶應義塾大学理工学部卒業．2007 年同大学大学院理工学研究科修士課程修了．同年同博士課程入学，現在に至る．ソフトウェア工学に関する研究に従事．

高田　眞吾（正会員）

1990 年慶應義塾大学理工学部卒業．1992 年同大学大学院理工学研究科修士課程修了．1995 年同博士課程修了．博士（工学）．同年奈良先端科学技術大学院大学情報科学研究科助手．1999 年慶應義塾大学理工学部情報工学科専任講師．2006 年より同大学助教授（現在は准教授）．ソフトウェア工学，情報検索等の研究に従事．電子情報通信学会，日本ソフトウェア科学会，ACM，IEEE CS 各会員．