4L-04

拡張画面遷移図に基づくWebアプリケーションのための テスト保守手法とそのフレームワーク

石上　椋一†　　高木　智彦‡

香川大学 創造工学部 創造工学科†‡

1. はじめに

Web アプリケーションの開発においては，利用者のニーズの変化や不具合などに迅速に対応することが求められる．そのため，Web アプリケーションは更新頻度が高く，それに伴うテスト資産の保守に要する労力も大きくなる傾向がある．このような問題に対応するために，たとえば，Web アプリケーションのテストコードを生成したり修正を支援したりできる手法やフレームワークなどが開発されている[1], [2].

本稿では，拡張画面遷移図と呼ばれる抽象化された形式的モデルに基づいて，Web アプリケーションのテストコードを保守する手法，およびそれを支援するフレームワークについて議論する．本手法は Model-Based Testing の応用であり，主にコードのレベルではなく，抽象的なモデルのレベルで保守を行うことで合理化を試みる点が特徴である．使用可能なテスト設計技法が機能テスト法に限定されるわけではないし，手作業でのテスト設計も許容される．ただし，本手法は，主に保守工程の改善を目的としたものであるが，Web アプリケーションの開発当初から導入される必要がある．本稿では，まず 2 節で本手法を提案する．また，我々は，Rust 言語で開発される Web アプリケーションを対象にした，本手法を支援するフレームワークを試作中であり，3 節でその概要を示す．最後に，4 節でまとめと今後の予定を述べる．

2. 提案手法

本研究で提案する手法は，以下の 4 つのステップから構成される．ステップ 1〜3 は新規開発工程，ステップ 4 は主に保守工程に対応する．

[ステップ 1] 開発者は，仕様に基づいて Web アプリケーションを実装する．その際，本フレームワークに含まれる，ソフトウェアアーキテクチャとして MVC モデルを採用したテンプレートなどを使用する．

[ステップ 2] 開発者は，本フレームワークに含まれる支援ツールを用いて，Web アプリケーションのソースコードから拡張画面遷移図を半自動生成する．拡張画面遷移図は，Web アプリケーションの画面に対応する節点，画面遷移に対応する弧，画面遷移のトリガとなる入力条件，画面遷移の直前／直後に Web アプリケーションが満たすべき条件（事前／事後条件），Web アプリケーションが常に満たすべき条件（不変条件）から構成される．支援ツールはソースコードを解析し，節点と弧を生成するとともに，拡張画面遷移図とソースコードとの間のトレーサビリティを確立する．その後，開発者は，仕様に基づいて節点と弧を接続し，入力条件や事前／事後条件，不変条件を追記して拡張画面遷移図を完成させる．

[ステップ 3] 開発者または支援ツールは，拡張画面遷移図上でテスト経路を指定する．そして支援ツールは，そのテスト経路に対応するテストコードを生成する．テストコードは，Web アプリケーションを実行し，HTTP メソッドの GET，POST，PUT，DELETE [3]の処理の正しさを事前／事後条件，不変条件に基づいて検証する．開発者は，支援ツールを使用してテスト実行ログを拡張画面遷移図上で俯瞰し，適宜個別分析する．バグを発見した場合，開発者は Web アプリケーションのソースコードを修正し，確認テストや回帰テストを行う．確認テストや回帰テスト用のテストコード（テスト経路）は，修正箇所に応じて既存のものの中から支援ツールにより選択され，実行される．加えて，開発者は，修正箇所についても拡張画面遷移図上で俯瞰し，テスト経路を適宜追加，修正，削除する．支援ツールはそれらをテストコードに反映し，実行する．

[ステップ 4] 仕様が更新された場合，開発者は，まず Web アプリケーションのソースコードを更新する．支援ツールは，更新されたソースコードを解析し，拡張画面遷移図の節点や弧を適宜追加，修正，削除するとともに，拡張画面遷移図とソースコードとの間のトレーサビリティを再確立する．次に，開発者は支援

Test Maintenance Technique and Its Framework for Web Applications Based on Extended Screen Transition Diagrams

† Ryoichi Ishigami・Faculty of Engineering and Design, Kagawa University

‡ Tomohiko Takagi・Faculty of Engineering and Design, Kagawa University

ツールを使用して拡張画面遷移図上でソースコードの更新箇所を俯瞰しながら，更新された仕様に基づいて拡張画面遷移図を更新する．そして，既存のテスト経路のうち，更新された拡張画面遷移図上で実行不可能なものを支援ツールにより抽出し，修正または削除する．開発者または支援ツールは，さらに，テスト経路を適宜追加した上で，再テストを行う．再テストによりバグを発見した場合の処置は，ステップ 3 と同様である．

3. フレームワークの試作

本研究において試作中のフレームワークの概要を図 1 に示す．本フレームワークの中核は，MVC モデルの基礎となるテンプレートと構成ファイル，および支援ツールであり，図において斜体で示されている．

Model はデータベース（DB）として実装される．SQL ファイルが，専用の構成ファイルに基づいて生成される．View はクライアントの表示を行う．Controller は，Web アプリケーションの各画面の GET，POST 処理やデータベースの更新，データの取得を行う．View と Controller はテンプレートを使用して実装される．テンプレートは，実装を省力化するだけでなく，拡張画面遷移図とソースコードとの間のトレーサビリティを確立する上でも重要な役割を担っている．

一方，支援ツールは，View と Controller を解析し，拡張画面遷移図の節点と弧を追加，修正，削除する．そして拡張画面遷移図やテスト経路の編集，弧を網羅するテスト経路の生成，テストコードの生成，選択，実行，更新箇所やテスト実行ログの拡張画面遷移図に基づく俯瞰表示などの機能を開発者に提供する予定である．なお，テストコードは，HTTP メソッドの検証コードと UI テストの自動化コードから構成される．前者は Rust 言語で，後者は Python 言語と Selenium [4]で実現される予定である．

4. おわりに

拡張画面遷移図に基づいて Web アプリケーションのテストコードを保守する手法，およびそれを支援するフレームワークについて議論した．我々の研究室に所属する学生数名が本フレームワークを試験的に使用したところ，小規模な Web アプリケーションと拡張画面遷移図の作成が可能であるとの見通しを得た．現時点で本フレームワークは未完成であり，加えて，テスト経路の生成アルゴリズムや，拡張画面遷移図の表記法なども改善の余地がある．今後の研究では，これらの検討と支援ツールの試作を進め，最終的にはバグの発見能力や労力などの観点で有効性を評価する予定である．

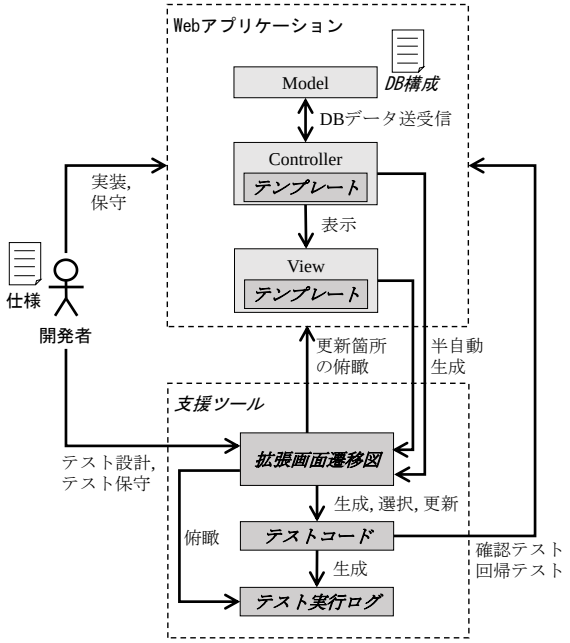

図 1: 試作中のフレームワークの概要（予定）

参考文献

[1] 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, Vol.56, No.3, pp.835-846, Mar. 2015.

[2] 中地祥剛, 崔恩瀞, 飯田元, 吉田則裕, "ウェブアプリケーション開発においての要求獲得のためのテスト記述支援環境の提案", 情報処理学会研究報告, Vol.2018-SE-200, No.6, pp.1-7, Nov. 2018.

[3] Jonathan Rasmusson, "初めての自動テスト-Web システムのための自動テスト基礎", オライリー・ジャパン, 2017.

[4] 伊藤望, 戸田広, 沖田邦夫, 宮田淳平, 長谷川淳, 清水直樹, Vishal Banthia, "Selenium 実践入門-自動化による継続的なブラウザテスト", 技術評論社, 2016.