# 画面操作を伴うテストにおける テストスクリプトの自動生成手法

倉林 利行[1,a)]　伊山 宗吉[1,b)]　切貫 弘之[1,c)]　丹野 治門[1,d)]

**概要**：近年，アプリケーションに対する変化のスピードが速いため，これらに対し，アプリケーションを短期間で対応させ，一定の品質を確保しつつリリースしていくことが強く求められている．短期間でリリースするためには，ソフトウェア開発全体の工数の 25%以上を占めているテストを自動化することが重要である．しかし，テストを自動化するために必要なテストスクリプトの作成に工数がかかるため，テスト自動実行ツールの利用率はわずか 12%とあるように，自動化が進んでいないのが現状である．本論文ではアプリケーションのソースコードと実行ファイルから静的解析と動的解析を用いてテストスクリプトを自動生成する手法を提案する．評価実験の結果，従来の人手でテストスクリプト作成する手法に対して，提案手法によって約 61%の工数を削減できることを示した．

# Automatic Test Script Generation on GUI Testing

Toshiyuki Kurabayashi[1,a)]　Muneyoshi Iyama[1,b)]　Hiroyuki Kirinuki[1,c)]　Haruto Tanno[1,d)]

## 1. はじめに

近年，アプリケーションに対するユーザのニーズの変化や，アプリケーションのプラットフォームとなるソフトウェア，ハードウェアの進化のスピードが速いため，これらに対し，アプリケーションを短期間で対応させ，一定の品質を確保しつつリリースしていくことが強く求められている．アプリケーションをリリースするにあたって，リリース前に適切に動作しているかどうかを確認するためのテストを行う．このとき，追加や変更があった機能をテストをするだけでなく，変更を加えていない，既存の機能のテストも実施する必要がある．既存の機能に対するテストを，回帰テストと呼ぶ．開発にかかる工数全体のうち，25%以上を回帰テストが占めている [1]．既存の機能の再テストという非生産的な回帰テストに工数が割かれることは，ユーザのニーズへ対応するための新機能の追加等の生産的な活動の妨げとなっており，回帰テストの工数削減が求められている．回帰テストの工数を削減するための方法として，テストの自動化が挙げられる．テスト自動化のためのツールとして，実際に画面を操作して動作を確認するテストを自動化する Selenium WebDriver[2] 等のテスト自動実行ツールが存在する．テスト自動実行ツールによって，テストにおいて実施したい画面操作等の手順をツールが読み取れる形式のテストスクリプトで記述することで，自動実行することができる．しかしテスト自動実行ツールの利用率はわずか 12%とあるように，導入が進んでいないのが現状である [1]．テスト自動実行ツールの導入が進んでいないことの理由として，テストを自動実行するために必要なテストスクリプトの作成に工数がかかることが挙げられる．テストスクリプトの作成では，テストの手順を逐一プログラムとして記述した後に動作確認や修正等も行う必要があり，同一のテストを人手で実行する場合より工数がかかる．そのため一般的には回帰テストを 3 回以上実行して初めてテスト自動化による工数削減効果が表れると言われている．しかし，回帰テストを 3 回以上実行する前に，仕様変更によってボタンや入力フォーム等の画面の操作対

[1] NTT ソフトウェアイノベーションセンタ, 東京都港区港南 2–13–34 NSS2 ビル 6F
[a)] kurabayashi.toshiyuki@lab.ntt.co.jp
[b)] iyama.muneyoshi@lab.ntt.co.jp
[c)] kirinuki.hiroyuki@lab.ntt.co.jp
[d)] tanno.haruto@lab.ntt.co.jp

象に変更があり作成したスクリプトが使用不可能になった場合，工数の元が取れなくなってしまう．Leotta ら [3] の調査によると，6 つのドメインが異なる OSS に対して人手でテストスクリプトを作成し，8 ヶ月以上期間をおいた後の新しいバージョンに対してテストスクリプトを実行したところ，61.2%のテストスクリプトに修正が必要だったことがわかった．このようにテストスクリプトに修正が必要となるケースの多さを考えると，テストスクリプトを作成するかどうかの判断は難しく，画面操作を伴うテストの自動化が進んでいない現状がある．

**本論文の狙いとスコープ**

本論文では，テストスクリプト作成にかかる工数を削減することで，テストを自動化し，回帰テストにかかる工数を削減することを目指す．本論文では 65.7%[1] と利用実績の多い Web アプリケーションをテスト対象として考える．また，画面遷移を網羅するテストスクリプトの作成を目標とする．

## 2. 既存研究

本章では，テストスクリプト作成の工数削減を目的とした既存研究について紹介する．テストスクリプトの作成を補助する技術として，ユーザの画面操作を記録し，スクリプトで出力するキャプチャ＆リプレイと呼ばれる手法が存在する．キャプチャ＆リプレイ手法の代表的なツールとして，Selenium IDE[4] 等が存在する．キャプチャ＆リプレイ手法では，GUI 操作でテストスクリプトを作成することができるため，テストスクリプトを直接プログラミングして作成する手法と比べ，プログラミングスキルのないユーザでもテストスクリプトを作成することができる一方で，保守性を意識しながら人が直接プログラミングして作成されたテストスクリプトと比較し，約 1.5 倍使用不可能になりやすく，メンテナンスコストが 19%〜104%増加してしまう [3]．また，ツール操作等の工数が発生するため，テストを人手で実行する場合と比較して工数が増えるという点は変わらず，回帰テストを 2〜10 回実施してはじめて工数削減の効果が表れると言われている [5]．したがって，テストスクリプトを作成するかどうかの判断が難しいという問題は解決できない．

丹野ら [6] は，入力フォームの値の条件や，物理 ID 等を含む画面設計書から，画面を遷移するのテストスクリプトを自動生成するモデルベーステストに基づく手法を提案している．本手法では，ウォーターフォール型開発のような，詳細な設計書を作成する開発現場では工数削減の点において効果的であるが，本論文が対象にしているような短期間でアプリケーションのリリースを繰り返す開発では，入力フォームの値の条件や，物理 ID 等を含む詳細な設計書が存在することは一般的でない．そのため，設計書を作成する追加の工数が発生するため，本論文の目的に合わない．一方で入力フォームの値の条件や，物理 ID 等を含む詳細なモデルではなく，業務フロー図等の抽象度の高いモデルからテストケースを生成するツールも存在する．例えば，stateMatrix[7] では，状態遷移表から N スイッチ [8] に基づくテストケースを自動生成することができる．また，TQCAssist[9] では，業務フロー図から業務フローを網羅するテストケースを自動生成することができる．いずれのツールも抽象度の高いモデルからテストケースが生成できるため，モデル作成の工数を削減できるといったメリットは存在するが，物理 ID 等のテストスクリプトに必要な情報をモデルが含んでいないため，テストスクリプトの自動生成はできない．

モデルではなくテスト対象からテストスクリプトを自動生成するための手法として，V. Dallmeier[10] らの研究が存在する．V. Dallmeier らの研究では，テスト対象のリンクやボタンを自動で押下していくことで画面遷移を抽出し，その画面遷移を実現するテストケースをテストスクリプトとして出力することができる．本研究では，ハイパーリンクやボタンのみで構成されたテスト対象に対しては，テストスクリプトを自動生成することができる．しかし遷移するための条件が存在する画面遷移を含むテスト対象に対しては，すべてのテストスクリプトを生成できないことがある．遷移するための条件が存在する画面遷移とは例えば，入力フォームに適切な値を入力することで発生する画面遷移や，DB が特定の状態の時に発生する画面遷移などが挙げられる．テスト対象にこのような画面遷移が存在する場合，対応するテストスクリプトは人手で補う必要がある．しかし，人手で補う箇所は仕様上の画面遷移図と出力されたテストスクリプトを比較しない限り明らかにならない．そのため仕様上の画面遷移図と生成されたテストスクリプトを 1 つ 1 つ対応付けて確認する工数と，足りないテストスクリプトを補う工数が発生してしまう．特に，画面遷移図と生成されたテストスクリプトを 1 つ 1 つ対応付けて確認するには，人がスクリプトを 1 つ 1 つ読み込んでどの画面遷移を通るスクリプトなのか調べていく必要があり，現実的ではない．しかしこの作業を怠ると，自動でテストスクリプトを生成できなかった画面遷移が不明なため，テストに抜け漏れが発生し，品質を落としてしまう可能性が生じるという問題がある．

## 3. 提案手法

提案手法では，工数削減とテスト品質を両立したまま，テストスクリプトを生成することで，回帰テストを自動化することを目指す．そのためには以下の 3 つの要求を満たす必要がある．

要求 1 事前準備やツール操作による追加の工数を必要とせず，テストスクリプトが自動生成できること
要求 2 自動生成できなかったテストスクリプトに対応する

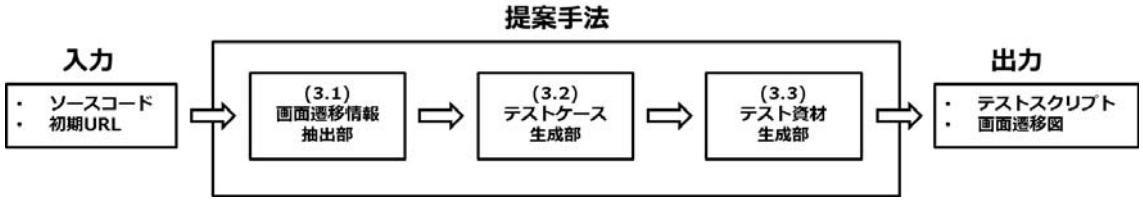


図 1　提案手法の全体像

画面遷移をユーザに提示することができること

要求 3 提示した画面遷移に対するテストスクリプトを人手で補助し，自動生成したテストスクリプトと合わせて実行することでテスト対象の画面遷移を網羅するテストができること

提案手法では，V. Dallmeier[10] らの研究における，「自動でテストスクリプトを生成できなかった箇所が不明なため，テストに抜け漏れが発生する可能性がある」という課題に対して，事前にソースコードの静的解析でテスト対象全体の画面構成を把握し，自動でテストスクリプトの生成ができなかった画面遷移をユーザに提示し，テストスクリプト作成を補助してもらうことで解決し，テストスクリプトを自動生成できるようにすることで，回帰テストの工数削減を実現する．提案手法では，ソースコードとテスト対象からテストスクリプトを自動生成するため，事前準備による追加の工数が発生しないため，要求 1 を満たす．また，テストスクリプトだけでなく，提案手法がテストスクリプトを自動生成できなかった画面遷移の情報を含んだ，テスト対象の画面遷移図を出力するため，要求 2 を満たす．出力された画面遷移図は，ソースコードの静的解析によってテスト対象の画面遷移の情報を漏れなく含んでいるため，要求 3 を満たす．以上の理由により，提案手法は上記 3 つの要求を満たすことで，品質を維持したままテストスクリプト作成にかかる工数を削減することができる．提案手法では，旧アプリケーションからテストスクリプトを生成し，生成したテストスクリプトをテストしたい新しいアプリケーションに対して実行することで，回帰テストの目的である以前から存在する機能が劣化していないかどうかを確認するという使い方を想定している．提案手法は，テスト対象のソースコードと初期 URL を入力とし，テストスクリプトと画面遷移図を出力とする．提案手法の全体像を図 1 に示す．提案手法における処理の流れは以下の通りである．

### 3.1　画面遷移情報抽出部

ソースコードから画面遷移情報を抽出し，確認すべき画面構成の全体像を把握する．例えば，Spring Framework[11] の場合，図 2 のように Controller に相当するクラス内の，さらにリクエストに応じるメソッド内の Return 文に遷移先が指示されているので，リクエストとそれに対応する遷移先を「画面遷移情報」として抽出する．

### 3.2　テストケース生成部

ユーザの与える初期 URL を始点としてテスト対象上のリンクやボタンを押下することで探索を行う．入力フォームが存在する場合は規定の値を入力する．各画面において，遷移先の画面が既に行ったことのある画面かどうか URL や HTML 等の情報を用いて判別する．新しい画面遷移の場合，そのときの手順をテストケースとして記録する．既に行ったことのある画面の数が (3.1) 画面遷移情報抽出部で取得した遷移先の数と一致しない場合，再度 1 つ前の画面に戻り探索を行う．一致する場合，もしくは対象画面の探索回数が一定数を超えた場合，その画面を解析済みの画面とし，解析してない別の画面に対して探索を行う．すべての画面の解析が終わるまで上記を繰り返す．

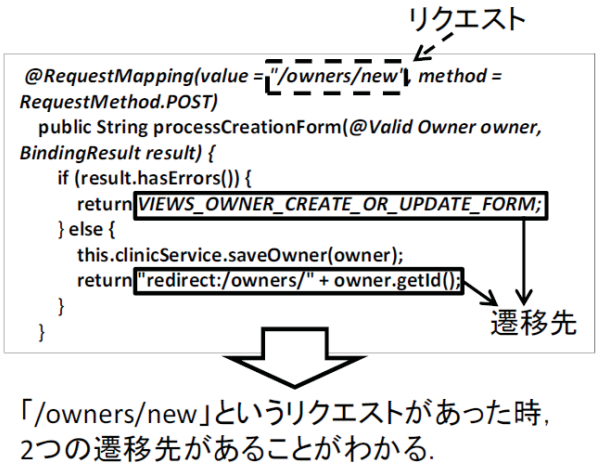


図 2　静的解析のイメージ

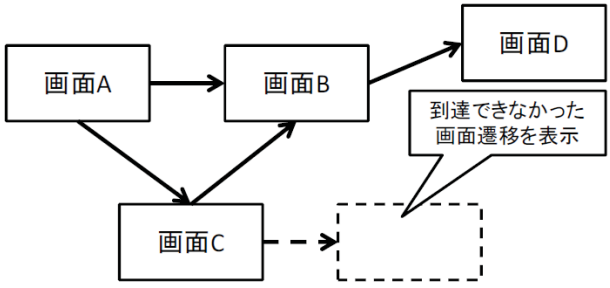


図 3　出力する画面遷移図のイメージ

### 3.3 テスト資材生成部

(3.2) テストケース生成部で記録したテストケースをテストスクリプトとして出力する．また，(3.1) 画面遷移情報抽出部で抽出された画面遷移に対応するテストケースが (3.2) テストケース生成部で生成できなかった場合，図 3 のように提示することでユーザへのテストスクリプトの補助を促す．

## 4. 評価

本章では，3 章で挙げた提案手法の 3 つの要求の評価を行った．テスト対象の題材として，SpringPetClinic[12] を用いた．

### 4.1 方法

#### 4.1.1 要求 1 に対して：工数の評価

従来手法の SeleniumIDE を用いてテストスクリプトを作成した場合と提案手法によるテストスクリプト生成にかかった工数を比較する．SeleniumIDE を用いる場合では，図 4 で示す SpringPetClinic の仕様上の画面遷移図を網羅するようにテストスクリプトを生成した時にかかった工数を測定した．なお，図 4 は画面が機能ごとに別れるように人手で作成した．提案手法を用いる場合では，初めに提案手法によってテストスクリプトと画面遷移図を生成し，その後ユーザが画面遷移図を見ながら自動で生成できなかった箇所の画面遷移に対するテストスクリプトを SeleniumIDE を用いて生成した．ユーザがテストスクリプトを生成する工数を測定した．

#### 4.1.2 要求２，３に対して：網羅率の評価

図 4 で示す SpringPetClinic の仕様上の画面遷移図に対する，提案手法が自動で出力したテストスクリプトの網羅率及び，人手でテストスクリプトを補った後の網羅率を評価した．

### 4.2 結果

#### 4.2.1 要求 1 に対して：工数の評価

従来手法の SeleniumIDE を用いてテストスクリプトを作成した場合と提案手法によるテストスクリプト生成のかかった工数の比較結果を表 1 に示す．提案手法では，従来手法に比べて，約 61%の工数を削減することに成功した．これは表 2 に示すように，提案手法が仕様書の画面遷移図に対して約 63%の網羅率のテストスクリプトを自動で生成することができたためである．また，提案手法を用いることによる追加の工数が発生しなかったため，仕様書の画面遷移図に対する網羅率と工数の削減率がほぼ同等な値になったと考えることができる．

#### 4.2.2 要求２，３に対して：網羅率の評価

表 2 に示すように，提案手法が仕様書の画面遷移図に対して約 63%の網羅率のテストスクリプトを自動で生成することができた．また，提案手法が出力した画面遷移図（図 5）における破線部分の画面に対する遷移，及びその先の画面遷移に対してテストスクリプトを人手で作成し，補った所，自動生成したテストスクリプトと合わせて仕様書の画面遷移図に対して 100%の網羅率となるテストスクリプトを生成することができた．これにより，提案手法はテストスクリプトをすべて自動で生成することができなくても，破線部分の画面に対する遷移，及びその先の画面遷移に対してテストスクリプトを人手で補うことで，漏れのないテストスクリプトが生成できることがわかった．

### 4.3 考察

提案手法では，従来手法（SeleniumIDE を用いたテストスクリプトの生成）に比べて，約 61%の工数を削減することに成功した．本節では結果の妥当性について，考察する．テストスクリプトを人手で作成することにおける課題は，同一のテストを人手で実行する場合よりスクリプトの作成に工数がかかるため，回帰テストを複数回実行しないと工数の元が取れず，今後の仕様変更等でテストスクリプトが使えなくなる可能性があることを考えると，スクリプト作成実施の判断が難しいことにあった．もし，人手でテストを実行する工数よりも，テストスクリプト作成の工数の方が少なければ，今後の仕様変更等でテストスクリプトが使えなくなっても，工数の元が取れないということはなくなるため，テストスクリプトを作成したほうが良いという判断を下せることになる．図 4 で示す SpringPetClinic の仕様上の画面遷移図を網羅するテストを人手で実行した所，440 人秒かかった．これは，今回提案手法でかかった工数である 393 人秒より，約 12%多い結果となる．したがって，今後の仕様変更等でテストスクリプトが使えなくなっても，提案手法でテストスクリプトを生成したほうが工数削減につながるということが言えるため，今回の結果は十分に妥当であると評価できる．

ただし一方で，今回の評価では画面遷移全体の 37%に対してテストスクリプトを自動生成することはできなかった．自動でテストスクリプトを生成できなかった原因は，２つが挙げられる．１つ目は DB に登録されたデータを検索し，検索結果を表示する画面遷移である．DB に登録されたデータと同一の入力値を生成することができなかったため，画面遷移することができなかった．２つ目は DB に新しくデータを登録し，登録結果画面に遷移する画面遷移である．提案手法では，入力フォームが存在した時，決められた一定の値を入力する．最初はその一定の値を登録することが出来たが，その後改めてその先に遷移する場合には，既に一定の値が登録されており，エラーとなり探索ができない．今後は画面遷移の条件に入力値や DB の状態を含む場合について対応し，テストスクリプトを自動生成できる範囲を広げたい．また，実システムを用いた評価を行

**表 1** 従来手法と提案手法の工数比較

| | 工数 [人秒] |
|---|---|
| 従来手法 | 998 |
| 提案手法 | 393 |

**表 2** 従来手法と提案手法の画面遷移網羅率比較

| | 画面遷移網羅率 [%] |
|---|---|
| 従来手法 | 100 |
| 提案手法（自動生成のみ） | 63 |
| 提案手法（自動生成＋人手サポート） | 100 |

い，妥当性を確認したい．


**図 4** SpringPetClinic の仕様上の画面遷移図

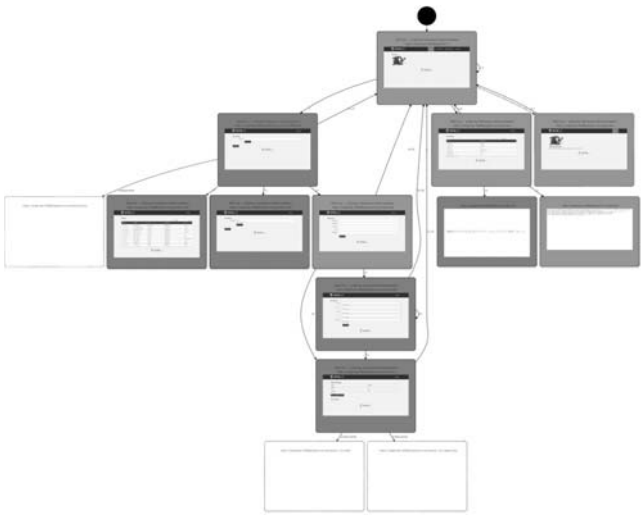

**図 5** 提案手法が出力した画面遷移図

## 5. まとめ

本研究では，テストスクリプト作成に工数がかかるという問題に対し，ソースコードとテスト対象からテストスクリプトを自動生成する手法を提案した．提案手法は以下の３つの要求を満たす．

要求１事前準備やツール操作による追加の工数を必要とせず，テストスクリプトが自動生成できること

要求２自動生成できなかったテストスクリプトに対応する画面遷移をユーザに提示することができること

要求３提示した画面遷移に対するテストスクリプトを人手で補助し，自動生成したテストスクリプトと合わせて実行することでテスト対象の画面遷移を網羅するテストができること

提案手法では，従来のキャプチャ＆リプレイ手法を用いたテストスクリプト生成手法と比較して，約 61%の工数を削減を実現することを評価によって確認した．今後は，画面遷移を実現するための適切な入力値の生成方法を検討することで，提案手法でテストスクリプトを生成できる範囲を広げ，より工数削減効果を高めていきたい．また，実案件を用いた評価も行い，実用化に向けた課題抽出を行いたい．

### 参考文献

[1] : ソフトウェアメトリックス調査 2012　報告書，http://www.juas.or.jp/servey/library/pdf/12swm.pdf.
[2] : Selenium WebDriver, http://www.seleniumhq.org/projects/webdriver/.
[3] Leotta, M., Clerissi, D., Ricca, F. and Tonella, P.: Capture-replay vs. programmable web testing: An empirical assessment during test case evolution, *2013 20th Working Conference on Reverse Engineering (WCRE)*, pp. 272–281 (online), DOI: 10.1109/WCRE.2013.6671302 (2013).
[4] : Selenium IDE, http://www.seleniumhq.org/projects/ide/.
[5] Fewster, M.: システムテスト自動化 標準ガイド，翔泳社 (2014).
[6] 丹野治門，　張暁晶：ソフトウェア設計ドキュメントを利用したテスト実行スクリプト生成技術の提案と評価，技術報告 16，NTT ソフトウェアイノベーションセンタ，NTT ソフトウェアイノベーションセンタ (2015).
[7] : stateMatrix, https://ja.osdn.net/projects/testools/.
[8] Beizer, B.: ソフトウェアテスト技法，日経 BP 社 (1994).
[9] : TQCAssist, http://www.xupper.com/lineup/option/tqcassist.html.
[10] Dallmeier, V., Pohl, B., Burger, M., Mirold, M. and Zeller, A.: WebMate: Web Application Test Generation in the Real World, *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, pp. 413–418 (online), DOI: 10.1109/ICSTW.2014.65 (2014).
[11] : Spring Framework, https://projects.spring.io/spring-framework/.
[12] : A sample Spring-based application, https://github.com/spring-projects/spring-petclinic.