Vol.2017-SE-195 No.26
2017/3/13

Webスクレイピングを用いた画面遷移図生成手法の提案

宍戸 成太[1]　上田 賀一[1]

概要：近年の Web 開発において，動的な画面生成は一般的な技術として普及しており，画面遷移の条件・UI は複雑化している．更に，Web 開発で主流なアジャイル開発では頻繁に機能追加・変更が行われ，同様に画面設計も変化する．画面遷移を記した設計書に画面遷移図があるが，手動での作成は非常に手間がかかり，保守もされにくい．そこで本研究では UI テスト実行と同時に画面遷移図を生成する手法を提案し，手法を実現するためのスクリプトを実装した．実際の Web サイトと自作の Web サイトに手法を適用し，不足なく画面遷移図を生成できることを検証した．

A Method of Generating Screen Transition Diagram by Using Web Scraping

Sisido Johta[1]　Ueda Yoshikazu[1]

Abstract: In recent web development, dynamic screen generation has become popular as a general technique, and the condition for screen transition and UI are complicated. Furthermore, agile development is in mainstream in web development, function addition and change are frequently performed, and the screen design also changes as well. Screen transition diagrams are described in design document, it is troublesome and difficult to maintain by hand. In this study, we propose a method to generate a screen transition diagram simultaneously with UI test execution, and implemented a script to realize the method. We applied the method to an actual Web site and our own Web site, and verified that we can generate a screen transition diagram without deficiencies.

Keywords: Web Scraping, UI test, Screen Transition Diagram, Agile Development

1. はじめに

近年の Web 開発では，動的な画面生成は一般的な技術として普及しており画面遷移の条件・ユーザインタフェースは複雑化している．画面遷移を記した設計書として画面遷移図がある．画面遷移図は画面とその移り変わりを図にした文書であり，アプリケーションの利用シナリオの確認，開発者間での対話に有用な文書である．しかし，画面遷移図は作成にコストがかかり保守されにくい．

Web 開発において主流な開発手法はアジャイルソフトウェア開発（以下，アジャイル開発）である．アジャイル開発は，イテレーションと呼ばれる短い期間単位を採用し，開発対象を細分化し，1 つのイテレーションで 1 つの機能というように追加的に機能を開発していく反復型開発である．多量の文書を作成することよりも，関係者間で必要な時に即座に顔を合わせて意思疎通を行うべきであること，実際に動くソフトウェアこそが最重要なプロジェクト進行尺度であることを強調しており [1]，他の開発手法と比較して成果物としての文書の量は少ないことが多い．

よって，アジャイル開発ではただでさえ文書が少ない上に，画面設計は複雑化し，短期間での変更が頻繁に行われるのに対し，それらを説明した文書が存在せず，新規にプロジェクトに参加した人材のソフトウェア理解が困難になることが懸念される．

[1] 茨城大学
Ibaraki University

© 2017 Information Processing Society of Japan

複雑化する画面遷移を全て網羅した画面遷移図を手動で作成し，頻繁に行われる UI 変更に合わせて保守することは非常に手間がかかる．

そこで本研究では，Web 開発における UI テスト実行と同時に画面遷移図を生成する手法を提案する．

2. 関連知識

2.1 UI テスト

UI テストとは，ユーザインタフェース（UI）に関わるテストのことであり，定義は様々であるが，本研究では実際に UI を操作して入出力結果の確認等を行うテストを想定している．

UI テスト自動化のための様々なツールが存在している．本研究では Selenium[2] を用いた UI テストを対象としている．

2.1.1 Selenium Builder

Selenium は，Web アプリケーションの機能テストや結合テストの自動化を実現するブラウザ駆動形のテストツール群であり，Selenium Builder はブラウザ上での操作を記録し，記録した操作をテストコードとしてエクスポート可能な Firefox のアドオンとして提供されている．Firefox のアドオンのため，対応しているブラウザは Firefox のみだが，テストコードのエクスポート形式が Java・Python など各種言語に対応している．本研究では，Python-unittest 形式でエクスポートしたテストコードを，改変元のテストコードとして利用している．

2.1.2 Selenium WebDriver

Selenium WebDriver（以降，WebDriver と呼称）は，Selenium が提供する Web ブラウザ操作のための API である．Selenium Builder がエクスポートするテストコードも WebDriver を用いて生成される．

2.2 画面遷移図

画面遷移図は，画面とその移り変わりを表した図である．画面同士を矢印等で繋ぎ，矢印の方向に画面が遷移することを表す．画面とその相互関係を一覧できる画面遷移図は利用シナリオの確認，顧客との打合せに有用であり，アプリケーション設計において重要な文書といえる．

画面遷移図を作成するツールはレイアウトを手動で整えるものと自動で整えるものに大別される．

2.2.1 レイアウトを手動で調整するツール

レイアウトを手動で整えるツールには以下の様なものが存在する．

Excel
図形を挿入して画面遷移図を作成できる．機能的には十分だが，元々は表計算ソフトであるため，図作成の用途以外の操作にも習熟する必要がある．

PowerPoint
Excel 同様図形を挿入して画面遷移図を作成できる．元々はプレゼンテーション用のツールのため，図作成の用途では不要な機能もあり，動作が重く，時間的コストもかかる．

astah [3]
UML モデリングツールだが，UML2.0 の状態遷移図（ステートマシン図）を画面遷移図として利用できる．

レイアウトを手動で調整するツールを使用するメリットは，画面名と遷移だけを表した粗い粒度のものから，ボタン等の配置や遷移先を含めた詳細なものまで描くことが可能である．

デメリットとして，作成のノウハウが開発者や組織の暗黙知に依存し属人性が高い．画面のレイアウト等の複雑な情報を扱う場合は編集コストが増大する [11]．更に，ファイルの形式から，マシンリーダブルでなく，構造が取り出せないため，バージョン管理にも不向きである．

以上のことから，これらのツールを用いて画面遷移図を作成するのは，保守にかかるコストが高いと言える．

2.2.2 レイアウトを自動調整するツール

レイアウトを自動で整えるツールには以下の様なものが存在する．

Graphviz [4]
dot 言語というスクリプトでグラフの内容を記述してビルドすることでグラフを画像として出力できる．

blockdiag [5]
Graphviz 同様スクリプトからグラフ画像を出力できる．

PlantUML [6]
シンプルな記述で UML を生成できるオープンソースプロジェクト．状態遷移図にも対応しているため，画面遷移図を作成できる．

レイアウトを自動調整するツールを使用するメリットは，ビルド元のテキストファイルがベースなので，バージョン管理にも対応し，レイアウトを調整する手間も無いため作成コストは低いことが挙げられる．

デメリットとして，UI のレイアウト情報等を含む詳細な図の作成には不向きである．更に，四角の形状変更，画像の挿入等の操作には独自のコマンドを使う必要があり，元となるテキストファイルが煩雑になりやすい．

2.2.3 UI Flows

UI Flows は，37signals[7] が考案した技法である．

図 1 に UI Flows の一例を示す．

具体的な UI 設計に触れず，「ユーザが見るもの」，「ユーザが行うこと」に注目することで，シンプルに UI の必要条件，あるいは骨子を記述することができる．必要な情報だけに絞ることで，情報の関係性，詳細が確認しやすく，図も小さく収まるという利点を得られる．

© 2017 Information Processing Society of Japan

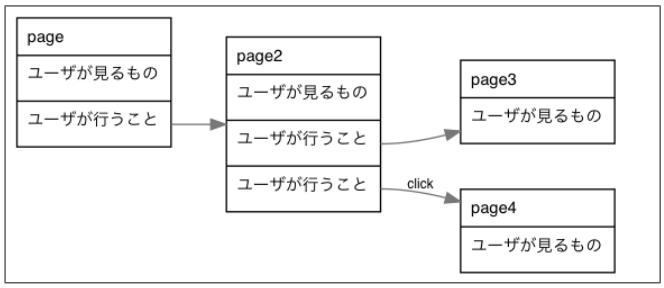

図 1　UI Flows の例
Fig. 1　Sample of UIFlows

2.2.4　uiflow

UI Flows を作成するツールとして，本研究では uiflow[8] を使用する．uiflow は，マークダウン様の記法（以降，uiflow 構文と呼称）で作成されたテキストファイルをビルドすることで，UI Flows 形式のグラフを出力するオープンソースのコンパイラである．図 1 を出力するための uiflow 構文を以下の**図 2** 示す．

```
[page]
ユーザが見るもの
_
ユーザが行うこと
==> page2

[page2]
ユーザが見るもの
_
ユーザが行うこと
==> page3
ユーザが行うこと
=={click}=> page4

[page3]
ユーザが見るもの

[page4]
ユーザが見るもの
```

図 2　uiflow 構文の例
Fig. 2　Sample of uiflow syntax.

uiflow 構文の記法によっては，図 1 の page2 から page4 への遷移のように矢印に名称をつけることも可能であるが，本研究では，遷移を起こすアクションは，ユーザが行うことをそのまま利用する．

2.3　Web スクレイピング

Web スクレイピングとは，インターネットを通じて Web サイトから情報を抽出するコンピュータソフトウェア技術のことである．インターネットから自動的にデータを収集するという分野の作業に対しては，スクリーンスクレイピング・データマイニング・Web ハーベスティング等の言葉が使われてきたが，今日では Web スクレイピングという単語を使うのが一般的な合意となっている [9]．

Web からデータを抽出し，抽出したデータを適切な形で利用するためのデータ分析，整形を行うため，Web スクレイピングは実際にはデータ分析，情報セキュリティといった幅広いプログラミング技術や技法に関わっていると言える．

2.3.1　BeautifulSoup

BeautifulSoup[10] は Python で動作する HTML と XML のパーサである．Python には標準で html.parser モジュールが提供されているが，Python のバージョンが古い場合，html.parser では正しくパースできない場合があるため，本研究では BeautifulSoup を使用している．Web スクレイピングに特化した Python のライブラリであり，Web ページから取得した HTML から，特定の要素を抽出するために，BeautifulSoup を使用することで簡潔に記述することができる．また，タグが閉じていない HTML を修正したり，パースした HTML に対して，コンテンツやタグの追加・除去等の操作を柔軟に行うことができる．

3.　提案手法

本研究では，Web アプリケーションにおける UI テストコードの実行で辿った画面遷移を記録し，画面遷移図を生成する手法を提案する．以下の 4 ステップで，テストコードから画面遷移図を作成する．

(1) テストコードの改変
(2) 改変後テストコードの実行
(3) uiflow 構文への変換
(4) UI Flows グラフの生成

この内，テストコードの改変，uiflow 構文への変換は自作のスクリプトを用いて行っている．

3.1　テストコードの改変

Selenium Builder で生成したテストコードは，ブラウザ上で行われた各操作を記録した順番でメソッドに変換しているのみで，各操作を個別に関数化し，関数を順次実行するという形式のものではないため，関数の実行前後で画面に変化があったかどうかを監視し必要な処理を挟み込むというアプローチを採ることができない．

そのため，本研究ではテストコードに直接改変を加えるというアプローチを採っている．

Selenium Builder で生成したテストコードに対し，画面遷移の記録を行い，DB に保存する処理を追加する．

テストコードは Python-unittest 形式でエクスポートされたものを対象としている．unittest は Python のユニットテストフレームワークであり，Python の標準モジュールとして提供されている．unittest では，テストの自動化・初期設定と終了処理の共有・テストの分類・テスト実行と結果レポートの分離などの機能を提供しており，以下のこ

とを行える．

- 各ユニットテストの前後に実行する setUp 関数，tearDown 関数を提供する．
- test_ から始まる全ての関数をユニットテストとして実行する（test_の接頭辞のない関数は無視する）．
- テストの成否を示す assert 文を提供する．（実行と結果レポートの分離）

テストコードの改変は元となるテストコードを 1 行ずつ読み取り，読みとった行内に特定の語句が出現しているかどうかを判断し，存在すれば直後に改変を追加している．改変は以下のステップに従う．

(1) 必要なモジュールのインポート
(2) setUp 関数内に DB とのコネクションの確立処理を追記
(3) test_ から始まる関数内にスクレイピング処理の追記
(4) tearDown 関数内に DB とのコネクション破棄処理を追記

test_ から始まる関数内に加えるスクレイピング処理は，画面遷移を取得するための機能を追加するものと，取得した画面遷移を DB に記録する機能を追加するものに分けられる．

3.1.1 画面遷移の取得

画面遷移を取得するため，テストコード中の以下の要素の出現を読み取り，スクレイピング処理を追加している．スクレイピング処理では，着目したい属性が変化したかどうかを監視し，変化があればクエリを作成し，DB へ記録する．

get() メソッド
get() は URL を指定してアクセスするメソッドであるため，ページ遷移が発生すると見做し，その直後にスクレイピング処理を挿入する．

クリック・ドラッグアンドドロップ
ユーザアクションが発生すると見做し，その直後にスクレイピング処理を挿入する．

ボタン押下
ユーザアクションが発生すると見做し，その直後にスクレイピング処理を挿入する．

3.2 DB への記録

画面遷移を記録する DB として，本研究では Oracle 社が提供している RDBMS の MySQL を使用している．DB へデータを記録するための事前準備として，以下に示す表 **表 1** の形式のテーブルを作成しておく．取得する情報として Unicode テキストを扱う場合は，MySQL はデフォルトでは Unicode を扱えないのでこの機能を有効にしておく必要がある．

elem は，スクレイピング処理で取得する着目したい属性が何かを表している．test_name は，改変後のテストコード実行時に取得したテストコードのファイル名を記録する．

3.3 改変後テストコードの実行

改変したテストコードを実行することで，自動で画面遷移を記録し，DB テーブルへ画面遷移情報を記録する．

3.4 uiflow 構文への変換

テーブルへの問い合わせから uiflow 構文へ変換する．page をページ番号，user_look を「ユーザが見るもの」，user_do を「ユーザが行うこと」として変換する．2.2.4 節で示したように，矢印に名称をつけることも可能であるが，遷移を起こすアクションは全て「ユーザが行う」ことを基本としている．

なお，ダブルクォーテーション，ハイフンなどの文字は，uiflow でのビルドの際にエラーの原因となるため，空白文字でエスケープしている．

3.5 UI Flows グラフの生成

生成された uiflow 構文のテキストファイルを uiflow でビルドすることで，UI Flows グラフが生成される．

3.6 差異検出

UI 変更によって，着目属性に変更があったかどうかを知りたいというユースケースを想定し，テストコード実行時に DB に格納されたテーブルの値から，遷移図間の差異を検出するスクリプトを作成した．遷移図間の差異の検出は，テーブルのテストコード名を指定し，テーブルに格納されているフィールドを比較して行う．

4. 適用事例

前章で示した手法とそれを実現するためのスクリプトを用いた画面遷移図作成を行い，手法の妥当性を検証した．本研究では Wikipedia と自作 Web サイトに対して実験した．

4.1 Wikipedia

Wikipedia に対する実験では，実際に存在する Web サイトに対して手法を適用可能かどうかを検証する．

Wikipedia の各記事ページの記事タイトルを着目したい属性（＝「ユーザが見るもの」）と定義する．記事タイトルは HTML ボディの h1 タグ内に記述されているため，スクレイピング処理で h1 タグの内容を取得する．

シナリオは以下に示す一連の操作を想定した．

(1) Wikipedia のメインページにアクセスする
(2) 検索ボックスに“大学”と入力し検索する
(3) “大学”の記事ページから “日本の高等教育” というリンクを探しクリックする
(4) “日本の高等教育”の記事ページから “大学院” と

表 1　DB テーブル構成
Table 1　DB Table Structure

id	page	user_look	user_do	elem	test_name
id(主キー)	ページ番号	ユーザが見るもの	ユーザが行うこと	属性	テストコード名

表 2　改変コード実行後のテーブル (Wikipedia)
Table 2　DBTable (Wikipedia)

page	user_look	user_do	elem	test_name
1	メインページ	send_keys(“大学”)	h1	Wikitest
2	大学	click()	h1	Wikitest
3	日本の高等教育	click()	h1	Wikitest
4	大学院	-	h1	Wikitest

いうリンクを探しクリックする

上記シナリオの実行を Selenium Builder で記録し，エクスポートしたテストコードをテストコード改変部に入力し，改変後テストコード実行後に作られた DB テーブルを**表 2** に示す（id の列は省略）．

表中の `send_keys("大学")` は入力ボックスに”大学”と文字入力する操作を行う WebDriver のメソッドである．表 2 の DB テーブルに対して uiflow 構文変換スクリプトを実行し，得られたテキストファイルを uiflow でビルドして得られた図を**図 3** に示す．

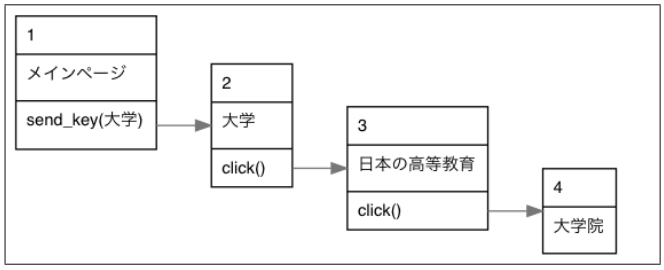

図 3　UI Flows グラフ（Wikipedia）
Fig. 3　UIFlows Graph （Wikipedia）.

4.2　自作 Web サイト

自作 Web サイトは本研究室で運用・管理している議事録管理システムであり，利用するのに利用者登録，ログインが必須となる．本節では，まず議事録管理システムに対し画面遷移図作成の実験を行い，その後，議事録管理システムの h1 要素に変更を行う前後の画面遷移の差異を検出する実験を行う．

4.2.1　画面遷移図作成実験

議事録管理システムに対し，以下に示すシナリオを想定した．

(1) 議事録システムのトップページにアクセスする．
(2) “ログイン”タブをクリックする．
(3) “ログイン”ページから “新規登録” というリンクをクリックする．
(4) “ユーザの追加”ページで 必要な項目を入力し “追加”ボタンをクリックする．
(5) “ログイン”ページに遷移するので，必要な項目を入力し “ログイン”ボタンをクリックする．

上記シナリオの実行を Selenium Builder で記録し，エクスポートしたテストコードをテストコード改変部に入力し，改変後テストコード実行後に作られた DB テーブルを**表 3** に示す（id の列は省略）．

“ユーザの追加”ページでは，ID（学籍番号）・名前・パスワード・メールアドレスの 4 つが必須入力項目となっており，“user_do”に入る項目が長大になったので途中省略している．

“ログイン”ページでは，ID・パスワードの 2 つが必須入力項目となっており，“ユーザの追加”ページ同様項目が長大になったので途中省略している．表 3 の DB テーブルに対して uiflow 構文変換スクリプトを実行し，得られたテキストファイルを uiflow でビルドして得られた図を**図 4** に示す．

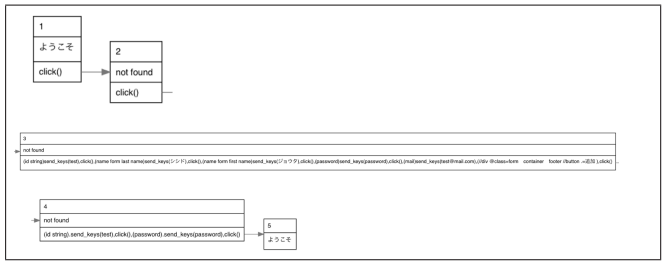

図 4　UI Flows グラフ （議事録管理システム）
Fig. 4　UIFlows Graph （Minutes System）

非常に横長の図として生成されてしまったため，手動で 3 分割した図を掲載している．

4.2.2　差異検出実験

本節では，議事録管理システムのトップページの h1 タグ内の要素を変更し，同じ内容のテストコードを実行した際の DB テーブルを比較し，変更を検出する実験を行う．

以下のシナリオを想定した．

(1) 議事録システムのログインページにアクセスする．
(2) 必要な項目を入力し “ログイン”ボタンをクリックする．
(3) トップページに遷移する

上記シナリオの実行を Selenium Builder で記録し，エクスポートしたテストコードをテストコード改変部に入力し，改変後テストコード実行後に作られた DB テーブルを**表 4** に示す（id の列は省略）．

© 2017 Information Processing Society of Japan

表 3　改変コード実行後のテーブル 議事録管理システム
Table 3　DBTable Minutes

page	user_look	user_do	elem	test_name
1	ようこそ	click()	h1	minutes01
2	not found	click()	h1	minutes01
3	not found	(id-string).send_keys(”test”) （中略）button[.=’追加’]”).click()	h1	minutes01
4	not found	(”id-string”).send_keys(”test”) (中略) button[.=’ログイン’]”).click()	h1	minutes01
5	ようこそ	-	h1	minutes01

表 4　改変コード実行後のテーブル （議事録管理システム 2a）
Table 4　DBTable （Minutes System 2a）

page	user_look	user_do	elem	test_name
1	not found	click()	h1	minutes02_01
2	ようこそ	click()	h1	minutes02_01

改変後テストコードを実行後，議事録管理システムの HTML の h1 タグ内を以下のように追加・変更した．

ログインページ

`<h1><center>ログイン</center></h1>` を追加

トップページ

h1 タグ内の　“ようこそ” → “Welcome!” に変更

HTML 変更後に改変後テストコードを実行した結果を**表 5** に示す（id の列は省略）．

表 5　改変コード実行後のテーブル （議事録管理システム 2b）
Table 5　DBTable （Minutes Sysytem 2b）

page	user_look	user_do	elem	test_name
1	ログイン	click()	h1	minutes02_02
2	Welcome!	click()	h1	minutes02_02

表 4 と表 5 が格納された DB テーブルに対して差異検出スクリプトを実行すると，page の値が同じ行同士を比較し，その内の値が異なるものが抽出され表示される．その実行結果を**表 6** に示す．

表 6　差異検出スクリプトの実行結果
Table 6　result of detecting diffrences

minutes02_01 page 1 user_look not found
minutes02_02 page 1 user_look ログイン
minutes02_01 page 2 user_look ようこそ
minutes02_02 page 2 user_look Welcome!

5. 考察

5.1 Wikipedia

Wikipedia に対する適用実験では，実際の Web ページに対して手法を用いて画面遷移図を適用できるかどうかを実験し確認した．実験の結果，メインページから“大学院”の記事ページまでの遷移を表した UI Flows グラフを生成することができた．図 4 では，最低限の情報のみで小さくまとまる図となっており，一覧性が高い．また，ビルド元がテキストファイルであるため，バージョン管理システムによる管理もしやすく，メンテナンス性も高い．高い工数が必要な画面遷移図の作成にかかる労力を軽減することが期待される．

しかし，次の画面への遷移が何をクリックすることで発生するのか不明であり，また，記事へのリンクがページ中の何処に存在しているかといったレイアウト情報が存在しないため，「何を目的とした図であるか」を共有できていない相手との対話に利用するのは難しいと思われる．また，「ユーザが行うこと」に改変前テストコード中の WebDriver のメソッド名をそのまま用いているため，開発者以外には意味が伝わりにくいことが懸念される．メソッド名を分かりやすい単語に変換する処理を実装し，変換前と変換後でどちらの文書が分かりやすいかを聞くといった評価実験を行う必要がある．

5.2 自作 Web サイト

5.2.1 画面遷移図生成実験

自作 Web サイトの議事録管理システムに対する画面遷移図生成実験では，議事録システムのユーザ追加からログインまでのシナリオを実行し，画面遷移図を生成できるかどうかを実験し確認した．uiflow では，「ユーザが行うこと」 を改行で複数並べることができないため，ユーザ追加画面における入力項目が多く，結果として，非常に横長の UI Flows グラフが出力される結果となった．出力結果そのままの状態では，電子的・物理的に扱うとしても 1 画面内での収まりが悪いため可読性が低く，UI Flows を利用するメリットが活かせていない．現状の uiflow の実装では，「ユーザが見るもの」の欄は改行に対応しているため，入力項目が複数存在する場合は省略するか，「ユーザが見るもの」 に入力項目も含めるなどレイアウトをより工夫する必要がある．

また，Wikipedia に対する実験における考察同様，「ユーザが行うこと」 にメソッド名をそのまま用いているため，開発者以外には意味が伝わりにくいことが懸念される．

しかし，DB テーブルに格納された内容に着目すると，ユーザが着目したい属性（h1 タグ内の要素）が記述されていないページであることを発見できており，不足している

情報と関連付けて，フロントエンドのリファクタリング方針を示すといった利用の方法が考えられる．

5.2.2 差異検出実験

差異検出実験では，議事録システムの h1 タグ内の要素を追記・変更した後で改変後テストコードを実行し，差異を検出できるかを実験し確認した．実際の Web 開発において，UI の変更は頻繁に行われることを想定し，着目属性に変更があったことを検出し提示することができた．

また，ログインページへの変更において，h1 タグ内に center タグを挿入し，レイアウト情報を含めた追記をしたが，center タグは除外され，タグ内の ”ログイン” のテキストのみを抽出できた．これは，BeautifulSoup が強力なスクレイピング能力を発揮し，着目属性とレイアウト情報を分離して取得できた例と見ることができる．

現状の実装ではコマンドライン上に検出した差異を表示しているだけであるため，今後は，UI Flows と組み合わせた視覚的な表示方法を考慮する必要がある．

6. 関連研究

横川らは，Web アプリケーションのテスト生成自動化を指向し Web ナビゲーション構造を UML ステートマシン図によってモデル化する手法を提案している [12]．作成されたモデルの検証によってアプリケーションの持つ誤りを正しく検知し，生成されたテストパターンは全てのナビゲーションを網羅したことが報告されている．

島田らは，過去の成果物の画面を機能ごとに分割し，それを再利用することで画面遷移図を作成する手法を提案している [13]．HTML ファイルと過去の成果物の遷移関係から画面遷移グループを作成し，DB に格納した情報から画面遷移グループを作成し，新システムの画面遷移図をゼロベースで作成する際の要求抽出にかかる作成や時間の削減，知識のない分析者の一助となることを目指している．

本研究では実際の画面遷移と同じ遷移を記した画面遷移図を生成するため，テスト実行と同時に辿った遷移を DB に記録する手法を提案した．島田らの手法とは HTML から得た情報を DB に格納する点で類似しているが，実際の画面遷移図を作成できる点で異なる．横川らの手法のように，生成した遷移図から，アプリケーションの誤り検出や実行したテストの妥当性の評価を行うことも，本研究の課題として考えられる．

7. まとめ

本研究では，Web アプリケーションを対象として，UI テストコードに改変を加え，テスト実行と同時に画面遷移を記録，記録した画面遷移を元に画面遷移図を作成する手法を提案した．また，手法を実現するためのスクリプトを作成し，手法の妥当性を検証した．適用事例では，Wikipedia と自作 Web サイトに対して実験を行った．自作 Web サイトでは，h1 タグを変更し，変更前後の差異検出の実験を行った．実験の結果，辿った遷移を過不足なく記述した UI Flows グラフを生成できた．

これにより，画面遷移図作成のためのコストを低減することが期待できる．

7.1 今後の課題

手法の適用実験の結果より，より視覚的に分かりやすい表示方法を考慮する必要がある．

また，本研究では単一のシナリオに沿った画面遷移図の生成を行ったが，実際の開発において，全体の画面遷移を一覧できる画面遷移図があれば，より遷移の抜け漏れ・設計の不具合の検出に役立つことが期待される．そのため，生成した遷移図同士を統合する機能の実装が必要である．

更に，本研究では手法を実現するためのスクリプトを実装したが，テストコード改変スクリプトの実行，改変後ソースコードの実行，uiflow 構文変換スクリプトの実行，uiflow でのビルドの各作業は手作業で行っている．今後は，これらの各作業を単一のツール内で行えるようツール化を目指したい．

参考文献

[1] Manifesto for Agile Software Development : http://agilemanifesto.org/
[2] Selenium : http://www.seleniumhq.org/
[3] astah : http://astah.change-vision.com/ja/
[4] Graphviz - Graph Visualization Software : http://www.graphviz.org/
[5] ブロック図生成ツール blockdiag : http://blockdiag.com/ja/blockdiag/
[6] PlantUML : http://plantuml.com/
[7] 37signals : http://37signals.com/
[8] uiflow : http://github.com/hirokidaichi/uiflow
[9] Ryan Mitchell: *Python* による *Web* スクレイピング, オライリー・ジャパン (2016).
[10] BeautifulSoupDocumentation : http://www.crummy.com/software/BeautifulSoup/bs4/doc/
[11] 秋山 浩一，奥村 有紀子，梅田 孝太，大友 貴司，坂間 洋平，浜田 浩史，松尾谷 恵 : WEB システムにおける画面遷移図表 表記法の提案と効果的なテストケースの作成，第五分科会（テスト設計グループ），(2008).
[12] 横川 智教，佐藤 洋一郎，有本 和民 : 検証およびテスト生成の自動化を指向した Web ナビゲーションのモデル化，情報処理学会，ウィンターワークショップ 2015・イン・宜野湾 論文集，pp.51-52(2015).
[13] 島田惇哉，榎本太紀，竹内享，秋吉政徳 : Web アプリケーションの送受信データを用いた画面遷移図作成支援手法，情報処理学会関西支部支部大会講演論文集 (2009).

© 2017 Information Processing Society of Japan