スマホ画面の回転に対応した WebVisulTesting ツールの提案

橋浦研究室　122D031 乙津 靖司　122D112 蓮見 璃空

1　はじめに

近年，スマートフォンや Web アプリケーション開発において Graphical User Interface (GUI) の品質確保は重要である．GUI の不具合はユーザの利便性を損なうため，変更後のレイアウトが仕様通りか検証するテスト工程は不可欠となる．しかし，手作業での確認は多大なコストを要する．既存の GUI 自動テストツール [1, 2] は多く存在するが，PC ブラウザを主対象としたものが多く，スマートフォンの画面回転（縦画面から横画面への切り替え）に伴う表示欠陥（画面サイズ調整不足による要素の重なりやはみ出し）を十分に検出できないという課題がある．そこで本研究では，スマートフォンの画面回転に対応し，レイアウトの変化を考慮した自動テストツールを提案する．

2　研究目的

この研究の目的は，スマートフォンの画面を回転することによる画面の表示欠陥 (画面サイズの調整不足) に対応する自動テストツールを作成することである．具体的には，HTML の動的要素に対応した自動テストツールである SpiderTailed2.0 [2] を拡張し，「スクリーンショットのリサイズ」,「比較ルール」という新たな機能を追加する．

3　提案手法

本研究では，SpiderTailed2.0 [2] を拡張し，「スクリーンショットのリサイズ」,「比較ルール」という機能を実装した自動テストツール SpiderTailed3.0 を開発した．レイアウト変換ルールにおいて縦画面から横画面への座標変換を行う際に，x 座標の予測には単回帰分析を，y 座標にはサポートベクター回帰 (SVR) を採用した．本ツールの実行手順を以下に示す．

1. Node.js [?] で Appium [3] サーバを建てた後に android エミュレータを起動する
2. 縦画面のスクリーンショットを取得
3. 画面回転
4. 横画面のスクリーンショットを取得
5. レイアウト変換ルールの適応
6. 判定ルールで比較
7. 比較結果の出力

なお，手順 1 は手動で行い，手順 2 から手順 7 までは本ツールにより自動的に実行される．この自動実行される処理の例を図 1 に示す．比較対象は HTML の `img` 要素とし，取得したスクリーンショットとレイアウト変換ルールを適用したスクリーンショットを比較する．以下の判定ルールを用いる．

- SIFT [4] による特徴点マッチングでマッチした特徴点数が 100 個以上の場合
- SIFT [4] による特徴点マッチングでマッチした特徴点数が 100 個未満かつ面積比が 89–113%に含まれる場合
- SIFT [4] による特徴点マッチングでマッチした特徴点数が 100 個未満かつ SSIM [5] による類似度が 0.7 を超えている場合

これらの条件のいずれにも該当しない場合，欠陥と判定する．比較結果の出力は HTML 形式のレポートで行う．レポートには，各 img 要素のセレクタ名と各判定ルールによる結果，総合評価が記載されている．

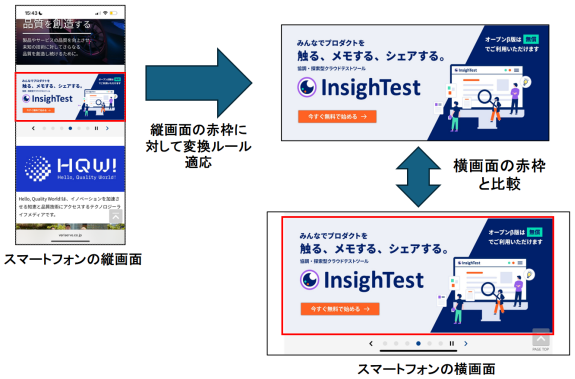

図 1: 手順 2 から手順 7 の内容例 [6]

4　実験

評価にはミューテーション解析を用いた．ミューテーション解析とはソフトウェアテストの品質を測る手法の 1 つである．具体的には，テスト対象とプログラムに対して人為的に誤りを挿入したプログラム (ミュータント) を生成し，ソフトウェアテストが挿入した誤りを検出できるかを調べる．

実験には，日本工業大学の就職実績のある企業一覧から，本ツール用に 21 社，SpiderTailed2.0 [2] 用に 4 社を選定し，実際の Web ページを対象としたミューテーション解析を実施した．本実験では，プログラムの誤りを 1 つ挿入した Web ページを本ツール用に 1050 個，SpiderTailed2.0 [2] 用に 550 個作成した．また，使用した端末は Pixel 9a，Pixel 9 Pro XL，Pixel 9 Pro の計 3 機種である．本実験で使用したミュータント一覧を表 1 に示す．

表 1: 本実験で使用するミュータント

#	誤りの種類	誤りの内容
1	テキスト	挿入
2	ファイル名	削除
3	width	10–200%
4	height	10–200%
5	hue-rotate	1–359 度
6	saturate	100%を除いた 0–200%
7	brightness	100%を除いた 0–200%
8	padding	10–100px
9	margin	10–100px
10	loading	lazy，eage
11	float	left，right，inline-start，inline-end

5　実験結果と考察

本研究で行った実験の結果を RQ ごとに以下にまとめる．

RQ1: 本ツールによる欠陥の検出は，挿入されたミュータントのタイプと関連があるか

$\mathcal{H}_0$ を「本ツールによる欠陥の検出は，挿入されたミュータントのタイプと関連がない」，$\mathcal{H}_1$ を「本ツールによる欠陥の検出は，挿入されたミュータントのタイプと関連がある」としてカイ 2 乗検定 (α =0.05, p=6.4295×10^{-28}) を行った結果，有意差が見られた．ミュータントの検出結果を表 2 に示す．具体的には，「brightness」という内側余白を変化させるミュータントに対しては高い検出能力を示した一方で，「hue-rotate」といった色味のみを変化させるミュータントの検出は困難であった．このような結果となった要因として SSIM の閾値設定が挙げられる．また，実験全体を通して 67 件の未検出が発生した．未検出が発生した要因として，以下の 2 点が考えられる．

1. 動的コンテンツによる要素取得の失敗
   具体的には，ボタン式スライドショーを含む箇所において，取得した画像のファイルサイズが 0 になるといった事例が発生し，画像比較を行うことができなかった．
2. 画像取得可否の不一致
   具体的には，ミュータント挿入前の正常な状態ではスクリーンショットの取得に失敗したが，ミュータント挿入後には取得に成功するといった現象が確認された．この場合，正解データが存在しないため，判定処理を実行できず未検出として扱われた．

表 2: ミュータント毎の検出可否分割表

#	ミュータント名	検出できた回数	検出できない回数	未対応	計
1	テキストの挿入	15	81	12	108
2	ファイル名の削除	16	82	3	101
3	loading	6	78	7	91
4	width	5	71	8	84
5	height	7	83	5	95
6	hue-rotate	0	92	3	95
7	saturate	16	75	6	97
8	brightness	49	49	6	104
9	padding	8	64	9	81
10	margin	2	94	3	99
11	float	2	88	5	95

RQ2: 本ツールの評価方法は SpiderTailed2.0 [2] と比較して，表示不具合と正解の検出精度をどの程度か

本ツールと SpiderTailed2.0 [2] において，再現率，適合率，特異率，F 値を算出した．本評価では，表示不具合検出精度の指標として特異率を，正解の検出精度の指標として F 値を採用した．

各指標の算出式を以下に示す．

- 再現率 = $\frac{\text{正解データとテストケースが } True \text{ の個数}}{\text{正解データが } True \text{ の個数}}$
- 適合率 = $\frac{\text{正解データとテストケースが } True \text{ の個数}}{\text{テストケースが } True \text{ の個数}}$
- 特異率 = $\frac{\text{正解データとテストケースが } False \text{ の個数}}{\text{テストケースが } False \text{ の個数}}$
- F 値 = $\frac{2*\text{再現率}*\text{適合率}}{\text{再現率}+\text{適合率}}$

算出結果を表 3 に示す．本ツールの正解検出精度は SpiderTailed2.0 [2] と比べて約 9.9%向上したが，本ツールの表示不具合検出精度は約 27.5%低下した．表示不具合の検出精度が低くなった要因として RQ1 で述べた通り，SSIM による「一致」と判定する閾値設定の影響が挙げられる．

表 3: 再現率，適合率，特異率，F 値の計算結果

#	対象名	再現率	適合率	特異率	F 値
1	SpiderTailed3.0	0.8200	0.9894	0.6786	0.8967
2	SpiderTailed2.0 [2]	0.8156	0.8156	0.9364	0.8156

結論として，本ツールは特定条件下の不具合検出に課題を残しつつも，正解に対する高い信頼性を確保しており，総合的な検出精度において先行研究 [2] を上回る性能を示した．

6 関連研究

Ren*et al.* [7] の研究では，モバイルアプリケーションの GUI 互換性テストにおける表示不具合を検出するための画像ベースの手法を提案している．

本研究では，レイアウト予測と画像評価によって欠陥検出を実現した．

7 まとめ

本研究では，スマートフォンの画面回転という動作に対応するテストツールの実装を行った．実験の結果，本ツールは不具合の検出精度を低下させてしまうが，正解の検出精度が向上することが分かった．

今後の課題として，未検出要因の 1 つであるボタン式スライドショー部分のスクリーンショットを取得できるようにするといった自動テストツールとしての安定性向上とツールの特異率向上があげられる．

参考文献

[1] AutoIt Consulting Ltd, “Autolt,” `https://www.autoitscript.com/site/`. (Accessed on 2025/4/11)

[2] 戸崎龍ノ介，平山宏人，HTML の動的要素に対応した GUI 自動テストツールの提案，卒業研究 (指導教員: 橋浦弘明)，日本工業大学，Jan. 2024. 〈`https://kbselab.com/pdfs/202403_R2.pdf`〉

[3] appium.io, “Appium,” `https://appium.io/docs/ja/2.0/`. (Accessed on 2025/2/12)

[4] David G Lowe, “Distinctive image features from scale-invariant keypoints,” International Journal of Computer Vision *(online)*, Vol.60, No.2, pp.91–110, 2004. (DOI:10.1023/B:VISI.0000029664.99615.94)

[5] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” IEEE Transactions on Image Processing *(online)*, Vol.13, No.4, pp.600–612, 2004. (DOI:10.1109/TIP.2003.819861)

[6] ベリサーブ株式会社，“ソフトウェアテスト · 第 3 者検証のベリサーブ,” `https://www.veriserve.co.jp/`. (Accessed on 2025/6/17)

[7] Yanwei Ren, Youda Gu, Zongqing Ma, Hualiang Zhu, and Fei Yin, “Cross-device difference detector for mobile application gui compatibility testing,” Proceedings of the 2022 IEEE International Conference on Software Testing, Verification and Validation Workshops(ICSTW) *(online)*, pp.253–260, Apr. 2022. (DOI:10.1109/ICSTW55395.2022.00052)