R-5

画像認識と DOM 解析を用いた Web Visual Testing 定量化手法の提案

橋浦研究室　　1175052　岡嶋 隆人　　1175146　竹田 浩紀

1. はじめに

昨今のアジャイル開発[1]の普及に伴い，反復的な Visual Testing による工数の増大が問題になっている．Visual Testing における見た目のバグを検知する作業は，人力によって行われているため，均質なテストを行うことは困難である．例えば，予め混入している間違いの数が分からない Web アプリケーションについて，全てのページに対して 1 ページずつ見た目の確認を行う場合，担当するエンジニアの能力や特性によって所要時間や見つけるバグの数が変動するという問題がある．このため，Visual Testing の所要時間に対する効果を定量的に評価することは困難である．加えて，ページの数に比例して作業量が増加する性質に対して，人的資源を増やさず作業時間を短くすることは難しいという問題がある．

そこで本研究では，見た目の仕様の数を定量化し，Visual Testing ツールを開発することで，テストの均質化と見た目のバグの定量化を図る．

1.1. Visual Testing とは

本研究で取り扱う Visual Testing とは，Web アプリケーションのフロントエンドに対して回帰的に行われるテストのことである[2]．Visual Testing は見た目のバグに関する迅速なフィードバックや早期検出，フロントエンド上の見た目の状態を保証し正常なコード状態でリファクタリングするために行われる[3]．そのような Visual Testing は図 1 のようなサイクルで実施される．

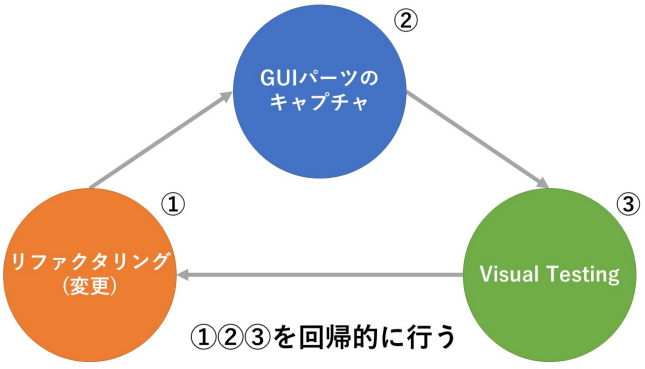

図 1：Visual Testing のサイクル

現状，人が目視で確認する方がツールより精度が高いため，未だに人力で行われている．しかしながら，見落としをなくすことはできないため，見た目のバグを網羅的に検知できない可能性が残っていた．

2. 研究目的

本研究の目的は，Visual Testing を支援するために見た目の仕様を定量化する方法を確立することである．このことにより，ツールによって均質なテストが可能となることを目指す．

3. 提案手法

均質な Visual Testing と見た目の仕様を定量化する方法を確立するにあたって，Visual Testing を行うツール SpiderTailed を作成する．本研究では本ツールの実装に際し，Sylvain ら[4]の研究を基に XML を用いて Web アプリケーションの見た目の仕様を表現すること(図 2)を提案する．

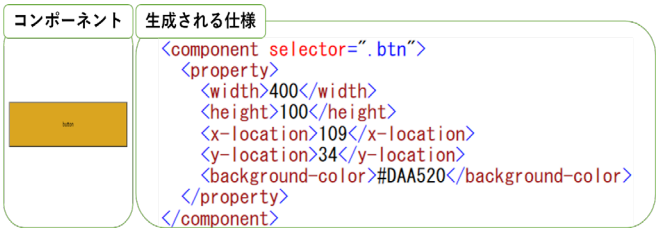

図 2：見た目の仕様の例

さらに，見た目の仕様を表現している XML のタグの数に着目し，property タグ内に出現するタグの数を見た目の仕様の数とみなすことで，見た目の仕様の数の定量化を図る．これにより，見た目の仕様を充足していない XML のタグの数を見た目のバグの数として集計することが可能となり，Visual Testing を定量的に評価することが可能となる．

Visual Testing の均質化を確認するにあたっては，モックアップと欠陥入りの Web アプリケーションに対する間違い探しを人力で行う場合と SpiderTailed を用いた場合に分けて行う実験によって評価する．

4. ツールの実装

SpiderTailed は HTML と CSS でフロントエンドが構成されている静的な Web アプリケーションを 1 ページ毎テストすることを想定する．実装する機能は以下の通りである．

1. 入力する HTML を解析する機能(F1)
2. 全 GUI パーツをキャプチャする機能(F2)
3. 見た目の仕様を生成する機能(F3)
4. 仕様検知機能(F4)
5. 実行結果を出力する機能(F5)

F2 の機能は Web ページ内の全ての GUI パーツをキャプチャする．

F3 の機能では F2 の機能で取得した画像から見た目の仕様として XML ファイルを生成し，F4 の機能で入力したもの同士で比較して検知する．F5 の機能では F1 から F4 の機能までの実行結果を提示する HTML レポート(図 3)を出力する．

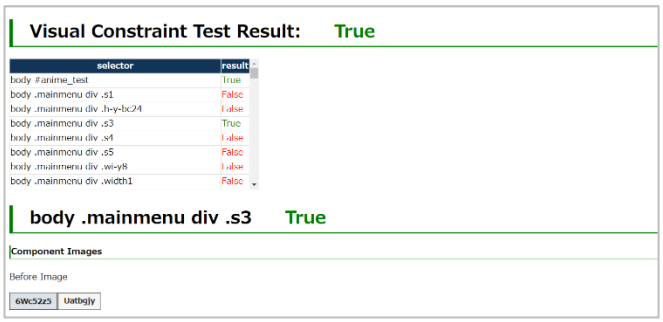

図 3：出力される HTML レポートの例

5. 実験

日本工業大学工学部情報工学科に所属する学生 8 名並びに先進工学部情報メディア工学科に所属する学生 6 名の計 14 名を被験者とする実験を行った．実験を行うにあたっては以下の 2 つの RQ を設定した．

1. ツールによる見た目の仕様の数の定量化
2. ツールによる Visual Testing の均質化

実験の概要は，被験者に間違い探しをしてもらう実験とツールで自動的に間違い探しを行う実験を行い，これらの結果を比較するものである．出題する問題は HTML と CSS で作成し，モックアップと欠陥入りの Web アプリケーションを 1 ページずつ用意した．また，被験者が間違い探しを行うにあたって，どのバグを見つけたのかを評価できるようにするため，予めモックアップの方を印刷した紙を配布し，見つけたバグに対応する場所に印を記入させた．

5.1. 人力による間違い探し

被験者が行う実験の流れは以下の通りである．

① 実験の概要・注意事項を被験者に説明する
② モックアップと欠陥入りの Web アプリケーションを被験者に配布する
③ Chrome を最大化して開いてもらう
④ 合図で間違い探しと時間計測を始める
⑤ バグを見つけたら紙の方に印をつける
⑥ 全て見つけ終えたと判断したら終了する

手順③では，ページ全体が把握できない状態でないかを確認する．また，1 つのモニターで表示しブラウザのタブの切り替えによって見た目のバグを探しもらう．加えて，被験者にはバグの数は教えないものとする．手順④では，合図で開始すると同時に所要時間の計測を始める．被験者が見た目のバグを全て見つけ終わったと判断するまで間違い探しを続けてもらい，手順⑤で被験者からその旨を宣言した時点で計測終了とする．その後，被験者から印をつけた紙を回収する．

5.2. ツールによる間違い探し

ツールで行う実験の手順は以下の通りである．

① 用意した問題をツールに入力する
② ツールを実行する

この実験の所要時間はツールに実装した処理の実行時間を合計した時間とする．実験結果は，出力される HTML レポートを確認することで把握する．

6. 評価方法

実験の評価は，実験結果を χ^2 乗検定や正解率(精度)，F 値の算出に使用し，その結果を分析することで行う．正解率及び F 値は以下の計算式で求められる．

$$\text{正解率(Accuracy)} = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{(式 1)}$$

$$\text{適合率(Precision)} = \frac{TP}{TP + FP} \quad \text{(式 2)}$$

$$\text{再現率(Recall)} = \frac{TP}{TP + FN} \quad \text{(式 3)}$$

$$\text{F 値(F-measure)} = \frac{2Recall \cdot Precision}{Recall + Presion} \quad \text{(式 4)}$$

7. 実験結果と考察

人力とツールで実験し，各検出を分類した結果を表 1 に示す．

表 1：バグの検出の分類

#		真陽性 (TP)	真陰性 (TN)	偽陽性 (FP)	偽陰性 (FN)
1	人力	835	769	1	47
2	ツール	63	20	35	0

比較すると，人力側では多くの偽陰性を検出したのに対し，ツール側では多くの偽陽性が見られたが偽陰性は見られなかった．また，実験で計測した所要時間を図 4 に示す．

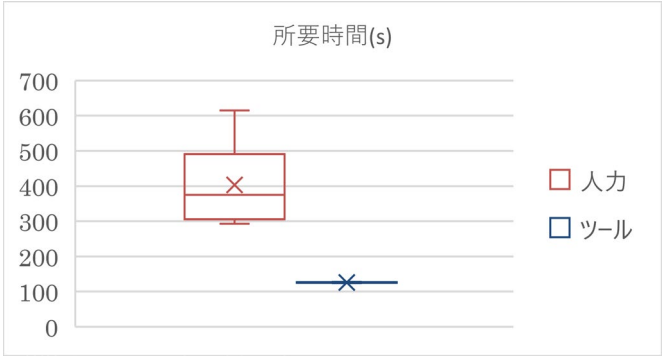

図 4：人力とツールの所要時間の違い

人力側では最大 5 分程度の差があるのに対し，ツール側では 2 分程度と一定だったことがわかる．

実験結果の独立性を確認するため，検定を行った．検定表の値は以下に示す式 5 と式 6 によって算出する．また，検定表及び検定結果は表 2 の通りである．

検知＝TP＋TN　　(式 5)

誤検知＝FP＋TN　　(式 6)

表 2：検定表と検定結果

#		検知	誤検知
1	人力	1604	48
2	ツール	83	35
3	χ^2乗値	176	
4	p 値	2.96×10^{-40}	

本検定では，帰無仮説H_0を「人力とツールの実験結果に独立性はない」，対立仮説H_1を「人力とツールのバグの実験結果に独立性はある」，有意水準αは 0.05 とした．表 2 より p<0.05 を満たすため，H_0を棄却しH_1を採用する．よって，今回の実験結果に独立性があることを確認した．

検定によって独立性が確認された表 1 の値から F 値を求めることによって，人力とツールの Visual Testing の精度の違いを確認する．算出した結果を表 3 に示す．

表 3：精度及び F 値の比較

#		正解率	適合率	再現率	F 値
1	人力	0.971	0.999	0.947	0.972
2	ツール	0.703	0.643	1.000	0.783

表 3 のデータを比較すると，人力側は高い精度で Visual Testing を行っていたが，全ての値において 1.000 を記録することはなかった．ツール側は，多くの偽陽性の影響で適合率や特異度が低くなってしまっているが，再現率において 1.000 を記録した．このことから，ツールは偽陽性を検出するが，漏れなく見た目のバグを検知可能であることが明らかになった．

最後に，これまでのデータから明らかになった人力とツールの Visual Testing の特性を比較した結果を表 4 に示す．

表 4：ツールと人力の特性の比較

#	特性	人力	ツール
1	誤検知の種類	偽陰性	偽陽性
2	所要時間の安定性	×	○
3	テスト品質の安定性	×	○

人力による Visual Testing とツールによる Visual Testing の特性を比較すると，相対するものであることが明らかになった．

前述したことを基に，RQ について回答する．RQ1 について，実験より本ツールは見た目の仕様を XML で表現し，見た目のバグの検知することで定量化できることを確認した．また，RQ2 について，本ツールには 2 つの利点があることを確認した．1 つ目は，表 3 の通り網羅的な検知が可能なことである．2 つ目は所要時間とテスト品質が安定していることである．したがって，ツールによって見た目の仕様の数を定量化し，均質に Visual Testing を行うことは可能である．

8. まとめと今後の課題

本研究の目的は，Visual Testing を支援するために見た目の仕様を定量化する方法を確立することであった．実験結果から，見た目の仕様を定量化でき，ツールによって Visual Testing を均質に行うことが可能であることを確認した．

今後の課題は 2 つある．1 つ目は，検知できるバグの種類を増加させることである．SpiderTailed で検知できる仕様の種類には限りあるため，ツールでテストできる Web アプリケーションには限りがある．今後はより多くのバグを検知できるようにする必要がある．2 つ目は，偽陽性の減少である．今回の実験では多くの偽陽性が検出され，ツールの精度に大きな影響を与えた．今後はこの偽陽性を減少させ，ツールの精度を改善していく必要がある．

謝辞

本研究を進めるにあたり，貴重な助言をいただいた新井啓之教授に感謝いたします．また，実験に協力してくださった日本工業大学の学生の皆さんに感謝いたします．

文　献

[1] L. Williams and A. Cockburn, "Agile Software Development: It's about Feedback and Change," Computer, IEEE, Vol. 36, No. 6, pp. 39-43, Jun. 2003.

[2] A. Issa, J. Sillito and V. Garousi, "Visual Testing of Graphical User Interfaces: An Exploratory Study Towards Systematic Definitions and Approaches," Proc. of 14th IEEE International Symposium on Web Systems Evolution (WSE 2012), pp. 11-15, Oct. 2012.

[3] Google LLC, "Android デベロッパー ドキュメント ガイド テストの基本," <https://developer.android.com/training/testing/fundamentals.html>, (Accessed 2020-03-15).

[4] H. Sylvain, B. Nicolas, G. Francis, L. B. Gabriel and B. Oussama, "Declarative Layout Constraints for Testing Web Applications, "Journal of Logical and Algebraic Methods in Programming, Vol. 85, pp.737-758, Aug. 2016.