Web GUIを対象としたOJSプロトタイプの実装と評価

岡嶋 隆人　橋浦 弘明

今日の Web やスマートフォンで利用されるアプリケーションにおいて，グラフィカルユーザインタフェース (GUI) は必要不可欠な要素である．GUI の外観はユーザの使用感だけではなく，商業的な成功にも影響を与えるとされている．このため，Web アプリケーションの実装を学ぶ上で，GUI に関する知識は必須となっている．こうした知識を身に着けるためには，学習者は講義や授業だけではなく，自主的な課外学習を行う必要があるが，既存のプログラム評価サービスは GUI を持つアプリケーションに対応していないことが多い．そこで，本研究では，Visual Testing 技術を用いることによって，Web ページの GUI を採点対象とするオンラインジャッジシステム (OJS) を実装した．また，実装した OJS のプロトタイプに対して，実際の学生による評価を行った結果について述べる．

1　はじめに

今日の Web やスマートフォンで利用されるアプリケーションにおいて，Graphical User Interface(以下，GUI) は必要不可欠な要素となっている．Pengnate ら [12] は，GUI のデザインはユーザがアプリケーションに対して抱く印象に影響を与え，そのことがアプリケーションの商業的な成功にも影響を与えると指摘している．よって，実用的なアプリケーションを開発する際には，GUI の実装知識が必要である．

このような知識を獲得するためには，学習者が講義や授業だけではなく自主学習を行うことができる環境が必要である．プログラミングの自主学習を行うための方法として，Online Judge System(以下，OJS) が挙げられる．OJS は国内外でサービスが開発・運用されており，代表的なものに paiza [11] がある．このようなサービスでは，システムの出題する課題に対し，学習者が解答としてプログラムを提出する．OJS は提出されたプログラムを自動で採点し，学習者にその結果をフィードバックするという機能を持つ．このため，学習者は時間と場所を選ばずに，自分のペースでプログラミング学習をできる．

しかしながら，こうしたサービスはプログラムの入出力として Command Line Interface(CLI) を期待していることが多く，今日の Web やスマートフォンで一般的に利用される GUI を持つプログラムに対応していないことが多かった．このような問題を解決するために，橋浦ら [14] は, VisualTesting 技術を利用した，GUI を持つプログラムを採点対象とした OJS を提案した．

本稿では前述の提案システムを実装し，学生の実際の使用を通じて評価を行った結果について述べる．

2　提案手法

始めに，本提案手法の概要を図 1 に示す．本提案手法は外部 CSS(`.css`) や JavaScript(`.js`) などを含まない単一の HTML ファイルで構成されている Web ページを採点対象としている．また，提案手法は 3 つの主要なコンポーネントにより構成されている．

1 つ目のコンポーネントは，ユーザである学習者とやり取りを行う画面群である．2 つ目のコンポー

---

* Implementation and Evaluation of OJS Prototypes for Grading Web GUI

Takato Okajima, Hiroaki Hashiura, 日本工業大学大学院工学研究科電子情報メディア工学専攻, Department of Electronics, Information and Media Engineering Major, Graduate School of Engineering, Nippon Institute of Technology.

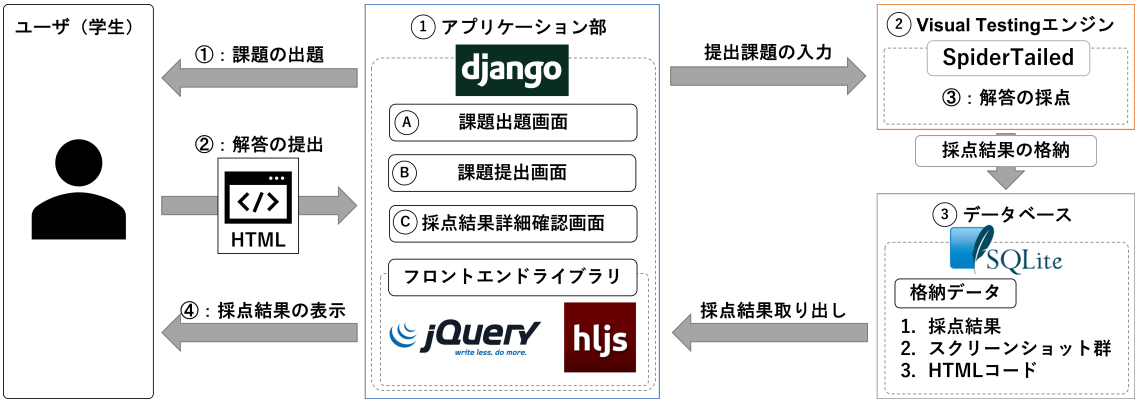

図 1　提案手法概要

ネントは，採点を行う VisualTesting エンジンである *SpiderTailed* [9] である．3 つ目のコンポーネントは，採点データ等を格納するデータベース (DB) である．

学習者が本提案手法を利用する流れを以下に示す．

1. 課題の選択
2. 解答の提出
3. 課題の採点
4. 採点結果の表示

2.1　課題の選択

本提案手法を用いた学習では，まず学習者は課題出題画面 (図 1-Ⓐ) にアクセスし，課題の一覧を確認する．そして，解答したい課題を選択することで課題提出画面 (図 1-Ⓑ) へと遷移する．課題提出画面には，課題として作成すべき Web ページ全体の画像と，ページ内の各要素の作成方法のヒントが掲載されている．

2.2　解答の提出

学習者は前述の情報を基に課題の解答となる Web ページ (HTML ファイル) を作成し，課題提出画面から提出 (アップロード) する．

2.3　課題の評価

解答が提出されると，OJS は事前に作成されている課題の模範解答となる Web ページをオラクル，解答として提出された Web ページをテスト対象として，*SpiderTailed* を用いてテストを実行し，課題の評価を行う．

評価が完了したら，結果と関連するデータを DB に格納する．本手法において，DB に格納するデータは以下の 4 つである．

1. 提出された Web ページ
2. Web ページの各要素のスクリーンショット
3. *SpiderTailed* により抽出された視覚的プロパティ
4. 各プロパティの比較結果

2.4　採点結果の表示

課題の採点終了後，学習者は課題提出画面の下部から，提出した解答の採点結果を確認することができる．詳細を確認したい採点結果を選択すると，採点結果詳細画面 (図 1-Ⓒ) に遷移し，学習者は以下の情報を確認することができる．

1. 各要素の採点結果
2. 模範解答と提出 Web ページの各要素のスクリーンショット
3. 各要素の視覚的プロパティの正誤 (模範解答と異なるものはハイライトされる)
4. 各要素に対応する提出 Web ページの HTML コード

3　実装

3.1　SpiderTailed とは

ここで，本研究が用いている VisualTesting エンジ

図 2　視覚的プロパティの抽出例 (キャプチャ画像)

表 1　視覚的プロパティの抽出例 (プロパティ)

#	プロパティ名	値
1	幅	800
2	高さ	150
3	相対 x 座標	10
4	相対 y 座標	10
5	背景色	#ADFF2F
6	文字色	#000000

ンである．*SpiderTailed* [9] について述べる．

SpiderTailed は，Web ページの各セレクタに対応する要素のキャプチャ画像から，幅や高さ，背景色といった視覚的プロパティを抽出し，抽出された視覚的プロパティをオラクルと要素毎に比較することで GUI のレイアウトの差異を検知するものである．

SpiderTailed による視覚的プロパティの抽出例を図 3.1 と表 1 に示す．図 3.1 のキャプチャ画像を入力した場合，*SpiderTailed* では表 1 に示すような視覚的プロパティが抽出される．

3.1.1　類似度によるマッチング

SpiderTailed はオラクルとテスト対象を HTML のレンダリング結果を比較し，それらが完全一致するかどうかを評価結果を出力している．一方，本手法は学習者が提出する Web ページと，模範解答の Web ページを比較する必要がある．例え仕様が同一であったとしても，教授者と学習者が完全に独立して Web ページの実装を行った場合，そのレンダリング結果が完全一致することは希であり，2 つの Web ページ間の構造が大きく異なることも十分起こりうる．したがって，これらの Web ページの HTML の構造に一定の差異があることを前提に，OJS として 2 つの Web ページが完全一致するかどうかではなく，それがどの程度一致しているかという一致の度合いを評価し，それを学習者にフィードバックしなければならない．

このため，著者らはオリジナルの *SpiderTailed* に対して，以下の 2 点の改良を加えた．

1. 要素のマッチングの実装
2. コサイン類似度による比較アルゴリズムの実装

1 つ目の要素のマッチングは，Web ページの HTML の構造に一定の差異がある場合に，2 つの Web ページ間で同等の要素同士を対応づける作業である．本手法ではオラクルとテスト対象それぞれの各要素のセレクタ同士のジャロ・ウィンクラー距離を計算し，最も類似したものを比較対象として対応づけることとした．

前述の通り，*SpiderTailed* は抽出した視覚的プロパティを要素毎に比較し，レンダリング結果が完全一致するかどうかを比較するものである．このような問題を解決するために，本研究ではコサイン類似度による比較を導入した．これにより，学習者に対して提出された Web ページが模範解答と一致するかどうかという結果だけでなく，模範解答とどの程度類似しているのかをフィードバックすることが可能になる．

本研究におけるコサイン類似度による採点例を図 4 に示す．まず，*SpiderTailed* で抽出された視覚的プロパティをベクトルに変換する．この際，値が設定されていないものに関しては，–255 という値を設定する．

変換したベクトルからオラクルとテスト対象の要素毎にコサイン類似度を算出し，これが予め設定した閾値を上回った場合，完全一致していなくとも，その要素を合格と判定する．そして，本手法では本採点アルゴリズムで，全要素に合格判定が出た場合，学習者がその課題に合格したものする．

3.2　システムの構成

提案システムは OJS という性質上，Web アプリケーションとして実装する必要がある．このため，Web アプリケーションフレームワークとして Django [6] を採用し，提案手法を基に以下の 3 つの要素に分けて実装を行った．

1. フロントエンド部
2. Visual Testing エンジン部
3. DB 部

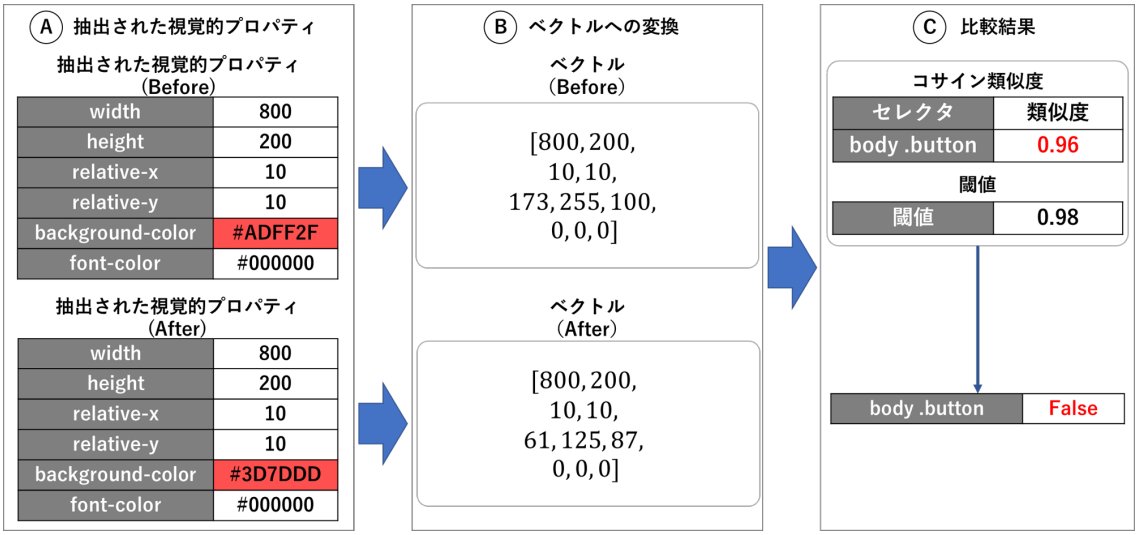

図 3　コサイン類似度による採点の例

3.2.1　フロントエンド部

フロントエンド部では，CSS フレームワークとして Bootstrap [2] を，ユーザとのやり取りのために jQuery [13] を，表示される HTML コードのシンタックスハイライトを行うために highlight.js [4] を用いた．

3.2.2　DB 部

DB 部では，Relational DataBase Management System(RDBMS) として，SQLite [5] を用いた．

3.2.3　Visual Testing エンジン部

Visual Testing エンジン部では，先に述べた改良を加えた，*SpiderTailed* を使用し実装を行った．ジャロ・ウィンクラー距離によるセレクタのマッチングは Python の文字列マッチングライブラリの 1 つである python-Levenshtein [1] を用い，コサイン類似度による採点アルゴリズムは，Python の行列計算ライブラリの 1 つである NumPy [10] を用いた．

4　評価

実装した OJS に対して実験による評価を行った．実験の概要は，本 OJS から実際に課題を出題し，その課題を学生に解いてもらう (Web ページを作成してもらう) というものである．

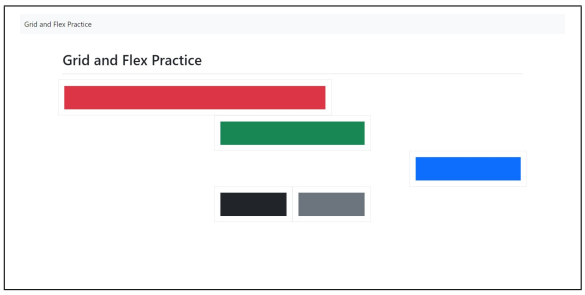

図 4　課題として出題した Web ページ

被験者は日本工業大学に所属する学部生，大学院生の計 11 名である．提出された解答に対し，提案手法と著者らの目視の採点を行い，それらの結果に対して考察を行った．

4.1　課題として出題される Web ページの作成

本実験において課題として出題された Web ページを図 4 に示す．課題として出題された Web ページは CSS として Bootstrap を使用して作成されている．表 2 に出題された Web ページの概要を示す．

本実験において，課題として出題される Web ページは Bootstrap を使用し作成されているため，被験者にも解答の Web ページ作成の際に Bootstrap を使用す

表 2　課題 Web ページの概要

#	項目	
1	LOC	59
2	要素数	20

るように指示した．また，被験者の開発環境の統一をするため，コードエディタとして VSCode [7] を使用するものとし，閲覧可能な Web サイトは，Bootstrap 公式ドキュメント [3] と MDN Web Docs [8] に限定し，解答に 45 分の制限時間を設けた．

4.2　RQ と評価基準の設定

本実験の評価の明確化のために，2 つの RQ を設定した．

RQ1　本手法の採点の目視との採点の一致率は？

RQ2　本手法の利用方法としてどのようなものが考えられるか？

RQ1 に解答するために，本実験において解答として提出された Web ページを著者らが目視で採点を行い，表 3 に示すように 4 種類に分類した．さらに，これらの分類結果を基に，正解率 (式 1)，適合率 (式 2)，再現率 (式 3)，特異度 (式 4)，F 値 (式 5) を計算し，その結果について考察を行った．

$$\text{正解率} = \frac{TP+TN}{TP+FP+TN+FN} \tag{1}$$

$$\text{適合率} = \frac{TP}{TP+FP} \tag{2}$$

$$\text{再現率} = \frac{TP}{TP+FN} \tag{3}$$

$$\text{特異度} = \frac{TN}{FP+TN} \tag{4}$$

$$\text{F 値} = \frac{2Recall \cdot Precision}{Recall + Precision} \tag{5}$$

RQ2 に解答するために，本実験における所要時間と，提出された課題のセレクタのマッチング結果について考察を行った．マッチング結果は提出された Web ページおけるマッチング結果を著者らが目視で確認を行い，類似した要素のセレクタにマッチしているならマッチング成功 (MS)，そうでない場合はマッチング失 (MF) とした．この時，提出された Web ページ内に対応する要素が実装されていないものは，集計対象外とした．

表 3　採点結果の分類表

		目視による採点結果	
		合格	不合格
OJS による採点結果	合格	真陽性 (TP)	偽陽性 (FP)
	不合格	偽陰性 (FN)	真陰性 (TN)

さらに，これらの結果から式 6 を基にマッチング率を算出した．

$$\text{マッチング率} = \frac{MS}{MS+MP} \tag{6}$$

5　実験結果と考察

本実験の各被験者の結果を表 4 に示す．

本実験では，被験者の全員が制限時間である 45 分を全て使用し，出題された課題に合格したものはいなかった．また，全体として，多くの偽陰性が検出された．

5.1　RQ1: 本手法と目視の採点の一致率

表 5 は，表 4 結果から総和を算出し，全体としての結果を求めたものである．

本実験では，正解率は 0.70 であり，半分以上の要素で目視との採点結果が一致した．また，適合率や再現性を見ると，適合率は上昇し，再現率は下降する傾向がある．これは，前述した偽陰性の影響を受けたためである．

これらの結果から，本システムは目視で不合格な要素を正しく判定することは得意だが，目視で合格である要素を合格とすることがは不得意であると言える．これは，*SpiderTailed* のテスト特性が，本システムにも現れてしまったことが原因である．

5.2　RQ2: 本手法の利用方法は？

表 4 を基に，総和を求め，マッチング率などを算出したものを表 6 に示す．

本実験では全体としてのマッチング率は 0.51 で

表 4　各被験者毎の実験結果

#	TP	TN	FP	FN	正解率	適合率	再現率	特異度	F 値	MS	MF	マッチング率
1	2	10	1	7	0.6	0.67	0.22	0.91	0.33	3	11	0.21
2	5	11	0	4	0.8	1.00	0.56	1.00	0.71	6	10	0.38
3	3	10	0	7	0.65	1.00	0.30	1.00	0.46	6	4	0.60
4	5	9	0	6	0.7	1.00	0.45	1.00	0.63	5	5	0.50
5	7	4	1	8	0.55	0.88	0.47	0.80	0.61	9	9	0.50
6	5	10	1	4	0.75	0.83	0.56	0.91	0.67	6	6	0.50
7	1	13	1	5	0.7	0.50	0.17	0.93	0.25	10	8	0.56
8	6	11	0	3	0.85	1.00	0.67	1.00	0.80	10	9	0.53
9	4	9	1	6	0.65	0.80	0.40	0.90	0.53	8	11	0.42
10	4	10	1	5	0.7	0.80	0.44	0.91	0.57	10	9	0.53
11	11	4	1	4	0.75	0.92	0.73	0.80	0.81	16	4	0.80

表 5　採点の一致率 (総和)

	TP	TN	FP	FN	正解率	適合率	再現率	特異度	F 値
総和	53	101	7	59	0.70	0.88	0.47	0.94	0.62

表 6　マッチング率 (総和)

	MS	MF	マッチング率
総和	89	86	0.51

あり，半分程度の要素でマッチングが失敗していた．マッチングの例として図 5.2 を示す．マッチングが成功した例では，課題の例と近しい見た目を持つ要素にマッチングしているが，失敗した例では，課題の例と全く異なる要素へとマッチングしてしまっている．

要素のマッチング率が低下した原因として，本手法では，セレクタの文字列としての類似度のみで要素のマッチングを行っており，視覚的な情報やセレクタの意味的な近さなど，他の特性を考慮していないことが考えられる．

これらの結果から，本研究における要素のマッチングアルゴリズムには，大きな課題が残っていると言える．

6　妥当性への脅威

本研究には，以下に述べる 3 つの妥当性への脅威がある．1 つ目は，評価実験における課題の採点である．本研究では，目視による課題の採点は，著者らが

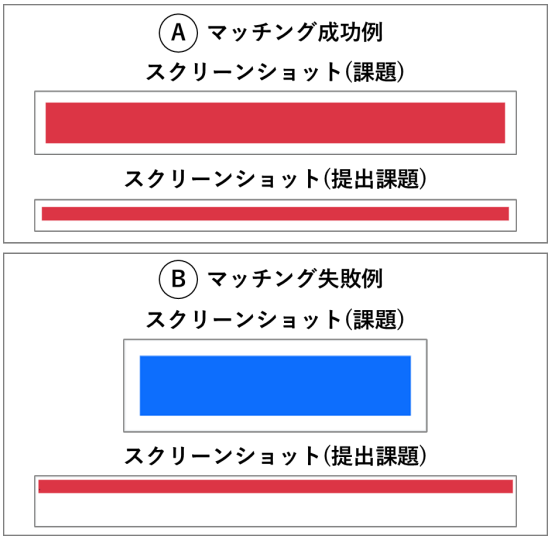

図 5　要素のマッチングの成功例と失敗例

行ったが，客観性の確保のために，実際に科目を担当している教員等に採点を行ってもらうなどして，その上で採点の一致度を算出する必要がある．

2 つ目は，出題された課題の Web ページである．本研究において課題として出題された Web ページは

Bootstrap を利用して作成されていた．しかしながら，実際の Web ページ作成では，こうしたフレームワークを使用しないこともあるため，評価結果の一般化のために，様々な実装形態を Web ページ課題として出題し，評価を行う必要がある．

3 つ目は被験者に関する問題である．本研究では，研究室内の学生を対象に実験を行ったが，より実際の講義に近い形式の実験を行う必要がある．具体的には，今後は講義の一部を使用し，実際に講義を受講している学生に本 OJS を使用してもらうことが考えられる．

7 結論

本研究は，VisualTesting 技術を利用した，Web ページの GUI の見た目を採点対象とした OJS を提案し，実際の学生の使用による評価実験を行った．実験の結果，現時点での本手法の実用性や課題が明らかになった．

今後の課題としては，以下の点が挙げられる．

採点アルゴリズムの見直し

本研究ではコサイン類似度を計測し，全ての要素で合格判定となった場合に，合格判定とするという方式であった．しかしながら，本実験で課題の合格者が 0 名だったということを踏まえると，採点方式を見直す必要がある．具体的には，課題 Web ページないに満たすべき最低限の要素を設定し，その要素全てに合格判定が出た場合に，その課題を合格とするものである．

課題設計の見直し

本研究の評価では，被験者に出題した課題は，Web ページの作成のみであったが，学生の理解度に応じた，段階的な課題を設計する必要がある．具体的には，CSS のプロパティ(`width` や `height` 等) の各項目毎に分け，学習者の理解の段階に応じた学習を行うことができるようにする必要がある．

動的な見た目の変化への対応

本研究で採点対象とした Web ページは，静的な見た目のものであったが，実際の Web ページでは，ハンバーガーメニューのように，マウスのクリックやマウスオーバーなどのアクションによって，見た目が変化する要素がある．今後はこのような動的な見た目の変化をする要素も採点対象にできるようにする必要がある．

謝辞 本研究の一部は JSPS 科研費 21K12179 の助成を受けた．

参 考 文 献

[1] Bachmann, M.: python-Levenshtein, `https://maxbachmann.github.io/Levenshtein/`, Accessed 2023-08-10.
[2] Bootstrap Core Team: Bootstrap, `https://getbootstrap.com/`, Accessed 2023-08-10.
[3] Bootstrap Core Team: はじめに - Bootstrap, `https://getbootstrap.jp/docs/5.0/getting-started/introduction/`, Accessed 2023-08-10.
[4] highlight.js: highlight.js, `https://highlightjs.org/`, Accessed 2023-08-10.
[5] Hipp, D. R.: SQLite, `https://sqlite.org/`, Accessed 2023-08-10.
[6] Holovaty, A., Willison, S., and Django Software Foundation: Django, `https://www.djangoproject.com/`. Accessed 2023-08-10.
[7] Microsoft Corporation: Visual Studio Code, `https://azure.microsoft.com/ja-jp/products/visual-studio-code/`, Accessed 2023-08-10.
[8] Mozilla: MDN Web Docs, `https://developer.mozilla.org/ja/`, Accessed 2023-08-10.
[9] Okajima, T., Tanaka, T., Hazeyama, A., and Hashiura, H.: SpiderTailed: A Tool for Detecting Presentation Failures Using Screenshots and DOM Extraction, *Knowledge-Based Software Engineering: 2022*, Cham, Springer International Publishing, 2023, pp. 39–51.
[10] Oliphant, T.: NumPy, `https://numpy.org/`, Accessed 2023-08-10.
[11] paiza 株式会社: paiza, `https://paiza.jp/`. Accessed 2023-08-10.
[12] Pengnate, S. F. and Sarathy, R.: An Experimental Investigation of the Influence of Website Emotional Design Features on Trust in Unfamiliar Online Vendors, *Computers in Human Behavior*, Vol. 67(2017), pp. 49–60.
[13] Resig, J.: jQuery, `https://jquery.com/`, Accessed 2023-08-10.
[14] 橋浦弘明, 岡嶋隆人, 田中昂文, 櫨山淳雄: Web アプリケーションに対するオンラインジャッジシステムの提案, 情報処理学会第 *84* 回全国大会講演論文集, Vol. 4(2022), pp. 397–398.