Research on Page Object Generation Approach for Web Application Testing

Yimei Chen, Zheng Li, Ruilian Zhao and Junxia Guo*
College of Information Science and Technology,
Beijing University of Chemical Technology, Beijing, China
*Corresponding: gjxia@mail.buct.edu.cn

***Abstract*—Test code generated by using the page object design pattern during web testing is easy to maintain. Page clustering is an essential stage of the page object approach. However, existing methods only consider the DOM structure in page clustering, which leads to inaccuracy when generating page objects. A state with the same DOM structure may result in an entirely different migration. The method of considering only the DOM structure cannot accurately generate page object classes. In order to improve the accuracy of page object generation, this paper not only considers DOM structure information but also considers CSS styles and the attributes of DOM elements in page clustering. Based on the experimental evaluation results, our method can automatically generate page objects that cover most of the application functions, which is more effective for the creation and maintenance of web test cases.**

***Index Terms*—Web applications, Automated testing, Page object.**

I. Introduction

As an essential stage of software development, software testing can be described as the process of identifying the correctness of programs. The primary purpose is to find possible errors in programs [1]. The cost of software testing is about 50% of entire software development cycle [2]. With the faster step of business, the software development cycle also becomes shorten and shorten. As a result, software testing needs to be finished in limited time, especially for quick-updated web applications. Because of cost constraints, software companies want to test their software as quickly as possible. Thus, research on automated testing has received significant attention in both industry and academia. Automated testing can run frequently, shorten the test cycle, quickly respond to changing requirements, and make the development process more agile.

The existing end-to-end automated test tools have a similar problem of maintaining test scripts during software evolution. Although those test tools can make testing easier, they cannot help testers write well-structured test scripts. To adapt to changing web applications, testers need to change test suites, which represents a considerable workload. Thus, the techniques for both test suite generation and maintenance are needed.

The page object pattern [3] is an effective mode on enhancing test suite maintenance and can reduce code duplication. A page object is an object-oriented class that acts as an interface to the web page of applications under test. Individually, all element attributes and element operations of a page object are encapsulated in a class. Whenever testers need to interact with elements of the user interface, the test case will use the approaches of the page object class. The test code is separated from the page elements and its methods of operation, in order to reduce the impact of changes in page elements on the test code. If page elements are changed, we only need to modify the code of the corresponding page object (the corresponding class) without modifying the test code. Therefore, it is beneficial to adopt the page object mode in maintaining web test suites.

However, the existing tools based on page object mode have their limitations. Only the DOM state is used for page clustering, which leads to the inaccuracy of generating page objects. Two web pages with the same DOM structure can show different visual effects and trigger completely different events, which should be divided into two page objects. To address this problem, we propose an approach for automatically generating page objects for web application testing, which is more accurate and efficient.

In this paper, we conduct in-depth research on page object generation and use it in web testing. Its main contributions are listed as follows.

1. Oriented to the automated generation of page object, we theoretically analyze influence factors related to page clustering. Based on the consideration of the DOM structure of two pages, we also consider CSS styles and attributes of DOM elements. According to the multiple influence factors, we propose a two-stage clustering algorithm.
2. We implement a prototype tool for automatically generating test cases for web applications, which combines multiple influence factors we propose in page clustering. In this way, we effectively solve the problems of repeated inaccurate testing in the automated testing process and improve the efficiency of the test.
3. We do a series of experiments with several kinds of web applications to evaluate the approach of this paper.

The organization of this paper is as follows. Section II presents the motivation. Section III introduces our approach, theoretically analyzing influence factors on page clustering and also proposes a clustering algorithm of multiple influence fac-

DOI reference number: 10.18293/SEKE2019-010

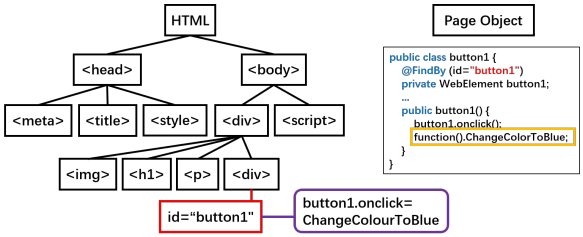

Fig. 1. A simple DOM tree of pageA.

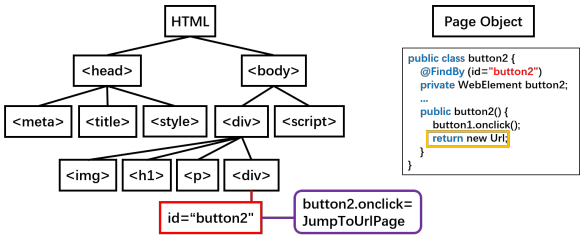

Fig. 2. A simple DOM tree of pageB.

tors. Section IV presents the experimental results to assess the effectiveness of our approach and its limitations. The relevant research is presented in Section V. Section VI concludes and presents the future works.

II. Motivation

Generally, generating page objects starts from a state-based model (graph) of the web application. The model consists of nodes and edges, where nodes are dynamic DOM states of web pages and edges are attribute-based transitions between nodes. The information of nodes and edges can be crawled from the web application. Then, the similar web pages are grouped into an abstract representation by the page clustering algorithm. Consequently, the related interactions between the web pages may change according to the clustering results. Finally, a set of page objects is generated for the abstract web pages and their interaction.

From the above analysis, we can find that the accuracy of page clustering is a crucial aspect when generating page objects. Page clustering relies on the similarity between web pages. However, the existing studies focus on structural features only, such as tag frequency, URL, DOM elements and so on, ignoring CSS styles and the attributes of DOM elements, to measure the similarity between web pages. This leads to the inaccuracy of page clustering and page object generation.

As an example, assume that two web pages with the same DOM structure can show different visual effects and trigger completely different events, which should be grouped into two page objects. However, the existing tools which only use the DOM structure information for clustering web pages will classify these two pages into one category, causing the inaccuracy of page objects. For instance, as shown in the Fig.1 and Fig.2, different attribute(i.e. id) values are set to the tags with the same DOM structure. Meanwhile, different attribute handlers $button1$ and $button2$ are set to

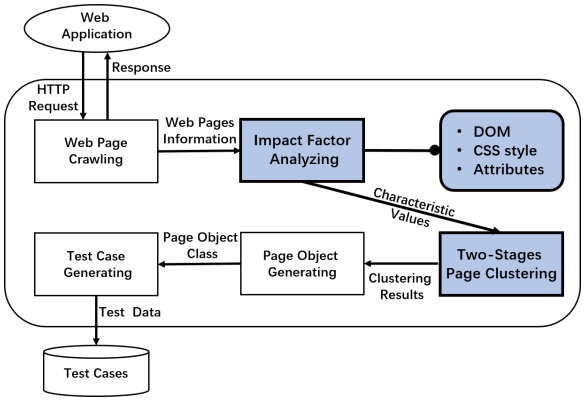

Fig. 3. An overview of the approach.

different ids. Even if their tags and structures are identical, pageA and pageB represent different states, and they should be divided into two page objects.

For measuring the similarity between web pages, the features which are using in existing approaches are insufficient. To solve this problem, we conduct an in-depth analysis of page information and raise a novel set of features considering the structure, CSS styles and attributes of pages, as well as the corresponding similarity measurement to distinguish pages with the same DOM structures but different functions.

III. Our Approach

A. Approach Overview

The overview of our approach is shown in Fig.3. This paper design an automated testing framework which has five main steps. In the first step, by using a specific crawler, we can simulate user actions to get information. Then, we theoretically analyze factors that affect page clustering, including DOM structures, CSS styles and attributes of DOM elements. In this step, we get their respective characteristic values. In step three, according to the two-stages clustering algorithm mentioned in this article, web pages are clustered. Next, page objects can be generated for every clustered page. Finally, test cases can be generated by calling page elements in corresponding page object classes.

B. influence factors related to page clustering

1) influence factors of DOM structure: DOM is a neutral interface between platform and language that allows programs or scripts to dynamically access content, styles and structure of updated documents [4]. The DOM structure represents the hierarchical structure of a web page and is very important for quantifying the similarity degree of web pages. In this paper, we use tree edit distance (TED) [5] to calculate the similarity of DOM. The Tree Edit Distance is the minimum number of tree editing operations that convert tree T to tree T'.

2) influence factors of CSS styles: In web applications, HTML tags not only display information about the structure and content of the page but also show some information about performance. Cascading Style Sheets (CSS) define how HTML

elements are displayed. Web developers nowadays put the information not only in the HTML tag hierarchy but also in the style sheets. CSS styles of web pages are also important information for page clustering.

In CSS, each style is usually named by a class value. The HTML elements in a web page are rendered by referring to the class values. The HTML elements with the same class value have the same style. Therefore, we consider using information on class attributes as a kind of assist for page clustering when generating page objects. This may avoid the misclustering case where the DOM structure of two pages is the same, but different CSS styles are used, which should not be classified into one category.

The process of building a style matrix between two pages is as follows. Firstly, we parse the HTML document of the web page and get a collection of all class values used in this page. We can record the class value information of the two web pages into two sets A and B. Then we can use Jaccard distance to calculate the distance between the class value matrix of two pages. The calculation method is listed in Equation 1.

$$D_j(A,B) = 1 - J(A,B) = \frac{A \triangle B}{|A| + |B| - |A \cap B|} \quad (1)$$

When $A = B$, $D_j(A,B) = 0$. The smaller the value, the more similar the two pages are. We use the calculation result of Jaccard distance as the style matrix values. According to the above steps, the style matrix values of any two pages can be respectively calculated and stored in the corresponding vectors.

3) influence factors of the attributes of DOM elements: If the structure of the two pages is the same, but the attributes of DOM elements bound on the nodes are different, different functions will be triggered. So the attributes of DOM elements may be helpful in clustering. HTML pages contain different HTML elements which may have different attributes and different functions. If all the attributes are taken into account, it will not only significantly increase the complexity of the method, but also reduce the accuracy of comparison, for example, the src attributes of <img> nodes.

The id value in a web page is a kind of unique identifier for a DOM node. Usually, scripting languages use id as a tag to find the node where id is located. So id value is significant for migration between states. We can distinguish different events bound to a node by id. Therefore, we consider that id value is a useful attribute in clustering and state migration.

We use Jaccard distance of id values to construct an attribute-based matrix between two pages. We get a collection of all the id values of a web page and use Equation 1 to calculate the id-distance of two web pages.

4) Using tag filter to reduce the DOM state: There are a large number of tags in HTML pages. Different tags have different effects. Not all tags in the HTML document are helpful in page clustering. We propose a tag filter method to reduce the pending tags in the HTML document, which focus on the effective tags and improves the usability and accuracy of the discovered tags. So that in page clustering, tag filter can remove some interference, which can improve accuracy and speed.

In HTML pages, tags like <head>, <style>, <script> and <link> contain data that is not displayed to the user as content, which is a factor in clustering with DOM structures. In theory, by removing these tags that are not used for structure analysis, not only will the DOM structure will become smaller and simplified, but also the consumption time will be reduced. DOM is the foundation of web application display. The more accurately the page structure is analyzed, the more completely its functions are understood. Based on the DOM structure, tag filter optimizes the clustering approach and indirectly improves the accuracy of subsequent operations.

C. A two-stages clustering algorithm using multiple influence factors

The factors affecting page clustering mentioned above have their pros and cons. They have different effects on different types of websites. If we only use one of them to measure all types of web pages, there will be inaccurate classification problems that affect the accuracy of page object generation.

Algorithm 1 A two-stages clustering algorithm using multiple influence factors

Input: HTML pages P_n crawled by crawler
Output: A set of clustering results S_j of HTML pages

```
Get P'_n by using tag filter on P_n
Calculate DOM tree edit distance matrix M_DOM of P'_n
Generate S_i using hierarchical clustering on M_DOM
n = number of crawled HTML pages in P'_n
i = number of clustering results in S_i
j = number of clustering results in S_j
for all (s_1, s_2, ..., s_i) ∈ S_i do
    if Consider CSS styles then
        Get class values sets S_CSS of P'_n
        Calculate M_CSS of S_CSS based on Equation 1
        Generate S_j using hierarchical clustering on M_CSS
        return S_j
    else if Consider attributes then
        Get id values sets S_id of P'_n
        Calculate M_id of S_id based on Equation 1
        Generate S_j using hierarchical clustering on M_id
        return S_j
    else if Consider CSS styles and attributes then
        Calculate M_Cid based on Equation 2
        Calculate M_Cid based on Equation 1
        Generate S_j using hierarchical clustering on M_id
        return S_j
    else
        return S_i
    end if
end for
```

Therefore, we propose a two-stage clustering algorithm using multiple influence factors as shown in Algorithm 1. We use tag filter on the pages crawled for preprocessing

TABLE I
TEST SUBJECTS.

Application	states	URLs	edges	DOM length
Aminer	12	4	17	58.714 kB
Termonline	57	3	56	26.194 kB
Musicbible	27	7	42	185.73 kB
BUCT	13	5	12	46.917 kB
DBLP	8	8	14	31.698 kB
CBA	17	17	51	31.156 kB
Chinacoop	55	49	108	18.26 kB
Xinxishibao	49	15	48	196.542 kB
Wanshifu	35	27	79	49.965 kB

to simplify the DOM structure. In the first stage, we use hierarchical clustering to cluster pages based on the DOM tree edit distance matrix of two pages. According to the clustering results of the first stage, we re-cluster the pages that are grouped into the same category. In the second stage, we consider different influence factors for clustering, CSS style and id attribute. Finally, the clustering results are returned according to different factors. This algorithm can correctly classify most different types of websites with better accuracy.

For any two web pages obtained by the crawler, the matrix M_{CSS} based on the influence factor of CSS styles can be obtained by calculating Jaccard distance of class value sets based on Equation 1 and the attribute matrix M_{id} can be gotten by computing Jaccard distance of id value sets. The final matrix value M_{Cid} of multiple influence factors according to a certain weight is calculated using Equation 2.

$$M_{Cid} = \omega \times M_{CSS} + (1 - \omega) \times M_{id} \tag{2}$$

ω and $(1-\omega)$ respectively represent the weight of the matrix value of CSS styles and the attributes of DOM ids. Setting reasonable weights in the final calculation is necessary. In order to determine the value of the above two weight parameters, this paper performs parameter adjustment experiments on several web applications.

IV. CASE STUDIES

In order to study the effectiveness of page object generation approach proposed in this paper, we selected thirty web applications in six fields as test objects, which contains the fields of information search, arts, portals, sports, governments and life services. Due to the limited space, we chose one or two of each category for display.In this section we list the result of nine web applications, including Aminer, Termonline, Musicbible, BUCT, DBLP, CBA, Chinacoop, Xinxishibao and Wanshifu. Then we use a two-stage clustering algorithm using multiple influence factors on these applications and analyze its impact on the accuracy of page object generation.

A. Test subjects

Table I gives the information about the nine experimental objects, including each application's name, the number of states, the number of visited URLs, the number of edges and the average length of the DOM.

TABLE II
COMPARISONS OF EIGENVALUES OF USING TAG FILTER OR NOT.

HTML	Tag Fliter	The Value of Eigenvector
index	N	0,731,0,697,719,734,742,772
	Y	0,727,0,693,715,730,738,766
state6	N	731,0,731,151,79,65,80,1195
	Y	727,0,727,151,79,65,80,1184
state14	N	697,151,697,0,128,145,147,1151
	Y	693,151,693,0,128,145,147,1140
state23	N	772,1195,772,1151,1183,1192,1211,0
	Y	766,1184,766,1140,1172,1181,1200,0

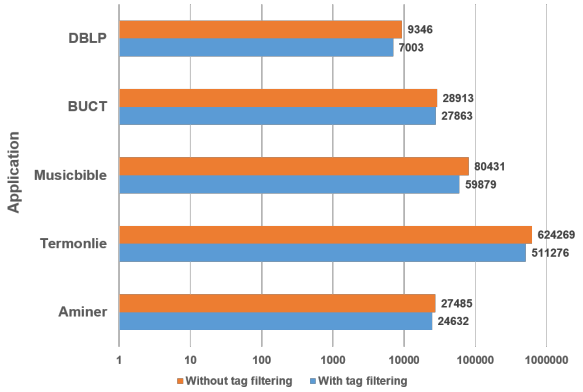

Fig. 4. Comparison of time consumption (in ms) in generating DOM-based feature matrix with or without tag filter.

B. Experimental results

1) Research on the effectiveness of tag filter on DOM states reduction: We compare page objects generated by using tag filter or not. We find that the page states generated before and after tag filter are the same.

Table II is the data of feature vector based on the DOM tree edit distance extracted from test subject DBLP. The first column shows the labels instead of different HTML pages. The second column shows whether tag filter is used to reduce DOM states. The remaining columns are the experimental value of the feature vector.

In Table II, the size of the number represents the degree of difference between DOMs. By observing experimental data, we find that tag filter can simplify DOM structure and make it more similarity.

Fig.4 shows a comparison of the time consumption of generating a DOM-based feature matrix using tag filter or not. It shows that the times using the tag filter are shorter.

Therefore, We can say that using tag filter will not only produce more reasonable page objects but also reduce time consumption. So we use tag filter to simplify DOM structure before generating a feature matrix.

2) Research on the effectiveness of two-stages clustering algorithm using multiple influence factors and weight setting: We compare generated page objects with different influence factors for test subjects and manually analyze page objects generated of each web application. The analysis results are used as criteria for judging the effects of each influence factor.

TABLE III
COMPARISON OF THE EFFECTS OF VARIOUS INFLUENCE FACTORS.

Application	DOM	DOM+CSS	DOM+ID
Aminer	1.52%	0.00%	0.00%
Termonline	0.88%	0.88%	0.00%
Musicbible	6.55%	6.55%	5.13%
BUCT	12.82%	0.00%	12.82%
DBLP	14.29%	0.00%	14.29%
CBA	13.97%	6.62%	8.82%
Chinacoop	14.41%	0.00%	5.39%
Epaperxxsb	0.00%	1.70%	1.70%
Wanshifu	2.52%	2.18%	1.01%
total	66.95%	17.93%	49.15%

Table III shows the result of the two-stage clustering algorithm proposed in this paper with the percentage of the mis-classified page combination number vs. the total page combination number. The first column is web applications under test. Second column shows the results that only using the DOM structure for page clustering. The third column shows the results that consider the DOM structure and CSS style for page clustering. The fourth column shows the results using DOM structure and attribute influence factors. The last line is the sum of the error rates for each method on all test subjects.

Through the data in Table III, we can find that for the application Aminer, the result of page objects generated when considering CSS style and DOM attribute for page clustering is the same as the result of the manual analysis. Moreover, they are both better than only considering the DOM structure. For the application Termonline, Musicbible and Wanshifu, the results that use DOM attribute have the lowest error rate when generating page objects. For the application BUCT, DBLP, CBA and Chinacoop, page objects generated by DOM structure and CSS styles have the lowest error rate. In addition to application Epaperxxsb, results of CSS-assisted DOM or ID-assisted DOM are better than using only DOM.

Therefore, we conclude that the generating accuracy of page object generated using our clustering algorithm is better than the method that only considering the DOM structure. However, the results of DOM structure with CSS styles and DOM structure with attribute have different performance for different applications. Therefore, we also do experiment that combine those two influence factors with different weights to achieve the best classification accuracy.

To set reasonable weights for CSS styles and DOM attributes in the final calculation, we did experimental research. In addition, we need to find the suitable condition for starting the second stage. If the result of first stage shows that the two pages are quite different, the second stage need not to be started. Therefore, We set the trigger threshold of the second stage in page clustering to be 0.5. When the distance between two pages exceeds the threshold in first stage, the clustering algorithm will start the second stage's processing.

The weights of the CSS style and DOM attribute values are set based on Equation 2. Then we raise the threshold of starting the second stage and adjust the weight of two influence factors. Through manual analysis, we get a reasonable threshold and weight set. The result of the manual analysis is used in our experiments.

TABLE IV
COMPARISON OF EFFECTS OF DIFFERENT WEIGHTS.

Threshold	ω=0.2	ω=0.4	ω=0.6	ω=0.8
0.5	6.55%	6.55%	6.55%	6.55%
0.6	6.55%	6.55%	6.55%	6.55%
0.7	6.55%	6.55%	6.55%	6.55%
0.8	6.55%	6.55%	6.55%	6.55%
0.9	6.55%	6.55%	6.55%	**3.13%**

The experimental results are shown in Table IV. Here we show the result of test subject Musicbible. The first column is the threshold of starting the second stage in page clustering. It is incremented from 0.5 to 0.9 to determine which weight has the best effect. The lines show the weight of CSS style influence factors based on Equation 2. The numbers in the table indicate the pages that are the percentage of the mis-classified page.

When the threshold of starting second stage is 0.9 and the respective weights of M_{CSS} and M_{id} are 0.8 and 0.2, the result of generating page object is most similar to the result of manual judgment results. Therefore, the best effect of two-stages clustering algorithm using multiple influence factors is to take 0.9 as the threshold of starting the second stage in page clustering, with the weight of o.8 for M_{CSS} and 0.2 for M_{id}.

V. RELATED WORKS

There are a number of testing approaches for web applications. Some of those approaches are suitable for the early days of Web applications, while others are more suitable for solving modern web technologies, such as AJAX and node.js [6], [7]. Some technologies are based on service-oriented framework, for example research [8]. Some use web page information extraction techniques. As a commonly used approach, the DOM similarity is used to judge the location of the required information, and the location information can be extracted directly without being disturbed by the noise information [9].

Web applications are different from traditional application software. web applications use BS structures and communicate with servers through browsers. However, local applications can run off the network. The difference is that web applications can only be run in various forms of browsers. In the traditional test [10], the web test is divided into two different test forms: black box and white box. Researchers use the corresponding models, such as control flow graphs or navigation maps, to conduct web application system testing studies. For example, Arcuri A [11] proposes a automated white-box testing method, in which test cases are generated automatically using evolutionary algorithms. Tests will be rewarded based on code coverage and fault finding metrics. Binkley D proposed a functional road map for testing web applications [12]. Liu X combined statement-based fault classification with spectrum-based software fault location in order to improve the accuracy of fault location and provide more possible fault information

for programmers [13]. In the research of Milani and Mirzaaghaei [14], by mining the human minds contained in the manually written test cases, they generate test cases for those points that are not found in the program under test. However, when the same function has multiple possible outcomes, a considerable amount of redundant test cases are generated. In the research of Yu Bing [15], a method based on a page object model for automatically generating web application testing is proposed. In this approach, the same function has multiple possible results, resulting in a significant probability that the generated page object is repeated, and finally, a redundant test case is generated.

When facing massive data information, it is possible to obtain incomplete or irrelevant information even through search engines, but with the development of web data mining, the problem is alleviated [16], [17]. Web data mining analyzes web page content to measure the importance of one page. Many algorithms have been proposed to extract content using the DOM tree. For example, in the method proposed in the research of Elyasov A [18], the DOM of the page is analyzed as an input to propose a testing framework of Javascript evolutionary. In the method proposed by Pagi V B [19], by using the embedded semantic tree kernel they can extract opinion content from web pages. In order to address repairing Internationalization Presentation Failures problems in web pages, Mahajan S [20] uses clustering to group stylistically similar elements in a page. It then performs a guided search to find suitable CSS fixes for the identified clusters.

Web pages contain a considerable amount of content that is not related to the web page theme. When acquiring information, this irrelevant information will affect efficiency and accuracy. It is often necessary to do some preprocessing on the web page. Uma [21] proposes an approach that cleans up and displays important information from web pages in standard format by eliminating noise and using unsupervised technology.

VI. Conclusions and Future Works

The existing generated page object tools only consider the DOM structure as the influence factor, which causes incorrect classification. In order to solve this problem, this paper theoretically analyzes the influence factors, which considers not only DOM structure, but also CSS styles and the attributes of DOM elements, and uses tag filter to simplify DOM structure. We propose a two-stage clustering algorithm using multiple influence factors when generating page objects. Therefore, the problem of incorrect classification of page objects is optimized. Moreover, we implement a prototype tool to automatically generate test cases for web applications to improve test efficiency based on page object. With experimental research, we find a suitable weight set for influence factors and threshold and verify the validity of our approach.

In future, we would like to analyze more factors. For example the type of events in client side scripts, which can handle different triggerable events. Moreover, in order to get more available information, we may use machine learning to determine whether state and function are related. Besides, we will extend tested objects into other types of applications.

Acknowledgment

The work described in this paper is supported by the National Natural Science Foundation of China under Grant No.61702029, No.61672085 and No.61872026.

References

[1] Offutt J, Ammann P.: Introduction to Software Testing. Cambridge University Press (2008)
[2] DENG Zhidan, YANG Haiyan, WU Ji.: Test data generation and selection approach for Web application based on constraint-solving. Computer Engineering and Applications **52**(18), 214–221(2016)
[3] PageObjects, http://code.google.com/p/selenium/wiki/PageObjects. Last accessed 13 Mar 2015
[4] W3C, https://www.w3.org/DOM/. Last accessed 1 Jun 2019
[5] Stefan Schwarz, Mateusz Pawlik, Nikolaus Augsten.: A New Perspective on the Tree Edit Distance. In: International Conference on Similarity Search and Applications 2017, LNCS, vol. 10609, pp. 156–170. Springer, Cham (2017).
[6] Serdar Dogana, Aysu Betin-Cana, Vahid Garousia.: Web application testing: A systematic literature review. Journal of Systems and Software **91**(1),174–201(2014)
[7] Wang Lina, Li Huai, Zhao Lei.: Ajax Web automatic testing model based on simulation of users. Journal of Huazhong University of Science and Technology (2016)
[8] Torsel A M.: A Testing Tool for Web Applications Using a Domain-Specific Modelling Language and the NuSMV Model Checker. In: Proceedings of the 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, pp. 383–390.(2013)
[9] Pan Xinyu, Chen Changfu, Liu Rong, Wang Meiqin.: Content extraction based on the similarity of the Web pages' DOM tree nodes path. Microcomputer and its Applications **35**(19), 74–77(2016)
[10] Myers G J, Sandler C, Badgett T.: The Art of Software Testing. 2nd edn.(2004)
[11] Arcuri A.: RESTful API Automated Test Case Generation. ACM Transactions on Software Engineering and Methodology (2019)
[12] Binkley D, Ceccato M, Harman M.: Tool-Supported Refactoring of Existing Object-Oriented Code into Aspects. IEEE Transactions on Software Engineering **32**(9), 698–717(2006)
[13] Liu X, Liu Y, Li Z.: Fault Classification Oriented Spectrum Based Fault Localization. Computer Software and Applications Conference. IEEE (2017)
[14] MilaniFard A, Mirzaaghaei M, Mesbah A.: Leveraging existing tests in automated test generation for web applications. In: ACM/IEEE International Conference on Automated Software Engineering. ACM, 2014, pp. 67–78.(2014)
[15] Yu B, Ma L, Zhang C.: Incremental Web Application Testing Using Page Object. In: Third IEEE Workshop on Hot Topics in Web Systems and Technologies. IEEE Computer Society, 2015, pp. 1–6.(2015)
[16] ZHANG Nai-Zhou, CAO Wei, LI Shi-Jun.: A Method Based on Node Density Segmentation and Label Propagation for Mining Web Page. Chinese Journal of Computers **38**(2), 349–364(2015)
[17] HUANG Yanjiao, WU Qin, LIANG Jiuzhen.: Boosted constrained conditional random fields for Web object information extraction. Computer Engineering and Applications **51**(23), 143–148(2015)
[18] Elyasov A, Prasetya I S W B, Hage J.: Search-Based Test Data Generation for JavaScript Functions that Interact with the DOM. In: Memphis, U.S.A, proceedings of the, IEEE 29th International Symposium on Software Reliability Engineering.(2018)
[19] Pagi V B , Wadawadagi R S.: Opinion Content Extraction from Web Pages Using Embedded Semantic Term Tree Kernels. International Conference on Computational Intelligence and Data Engineering, pp. 345–358(2018)
[20] Mahajan S, Alameer A, Mcminn P.: Automated Repair of Internationalization Presentation Failures in Web Pages Using Style Similarity Clustering and Search-Based Techniques. In: 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST). IEEE Computer Society.(2018)
[21] Uma R, Latha B.: Noise elimination from web pages for efficacious information retrieval. Cluster Computing, pp. 1-20(2018)