Software Impacts 17 (2023) 100516

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

BEAUD: A Browser Extension to Automatize End-User Deeds

Iñigo Aldalur

Mondragón Unibertsitatea, Loramendi 4, Mondragón, 20500, Spain

ARTICLE INFO

Keywords:
Web Augmentation
Browser Extension
Automation

ABSTRACT

Web Augmentation allows anyone, computer-literate or not, to adapt the content, style or behavior of any web page. Searching for information is not always an undemanding task. The information searches carried out by users cause frustration when they are carried out frequently. Even more so when the interactions accomplished are numerous or when these interactions lead to errors. In this paper, we present BEAUD, a Web Augmentation extension that allows automating the information search processes that users make on the web. This tool drastically reduces the interactions that users make to the minimum, reducing errors and frustration.

Code metadata

Current code version	0.2.3
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2023-132
Permanent link to Reproducible Capsule	
Legal Code License	GPL
Code versioning system used	Git
Software code languages, tools, and services used	JavaScript, HTML and CSS
Compilation requirements, operating environments & dependencies	Google Chrome and Microsoft Edge
If available Link to developer documentation/manual	https://github.com/ialdalur1/BEAUD/blob/main/README.md
Support email for questions	ialdalur@mondragon.edu

1. Motivation and significance

The need to adapt Web pages to the needs and desires of users has grown in recent years. To carry out this task, different techniques can be used: personalization, customization or Web Augmentation (WA).

Web personalization is defined as ”any action that adapts the information or services provided by a Website to the needs of a particular user or a set of users, taking advantage of the knowledge gained from the users' navigational behavior and individual interests, in combination with the content and the structure of the Website” [1]. The objective of a Web personalization system is to “provide users with the information they want or need, without expecting from them to ask for it explicitly” [2]. On the other hand, in customization, the web page is adapted to each user's desires or preferences with regard to its structure and presentation. Whenever a user logs in, his customized web elements are loaded [1]. However, in WA, the user does not need to be logged in, this technique permits adapting any website to the user preferences. Web augmentation tools ”modify Web pages, either to insert interfaces of their own or to add structure (e.g. links) to the Web page. This modification can take place at the Web server (perhaps a Web server translating a proprietary data format into HTML), by calling scripts that return modified pages, by using a special proxy, or by modifying the Web pages as, or after they are displayed in the Web browser” [3]. Dı ıaz, Arellano and Azanza said that ”WA is to the Web what Augmented Reality is to the physical world: layering relevant content/layout/navigation over the existing Web to enhance the user experience” [4].

The first two techniques (personalization and customization) can only be performed by being the owner of the site [1]. The WA allows adapting any web page independently of the creator. This feature of the WA is very important so that any user can adapt any web page without the need of the owner. Web pages do not always provide all the information needed by the user. At other times, the desired information is not visible and the user must interact with the page to find the desired information. The most common actions are clicking, scrolling, opening a new tab, typing, etc.

The performance of these actions causes users to suffer frustration [5]. In addition, it should be noted that each of the actions that a user can perform when searching for information on the web

E-mail address: ialdalur@mondragon.edu.

https://doi.org/10.1016/j.simpa.2023.100516
Received 18 April 2023; Received in revised form 9 May 2023; Accepted 17 May 2023

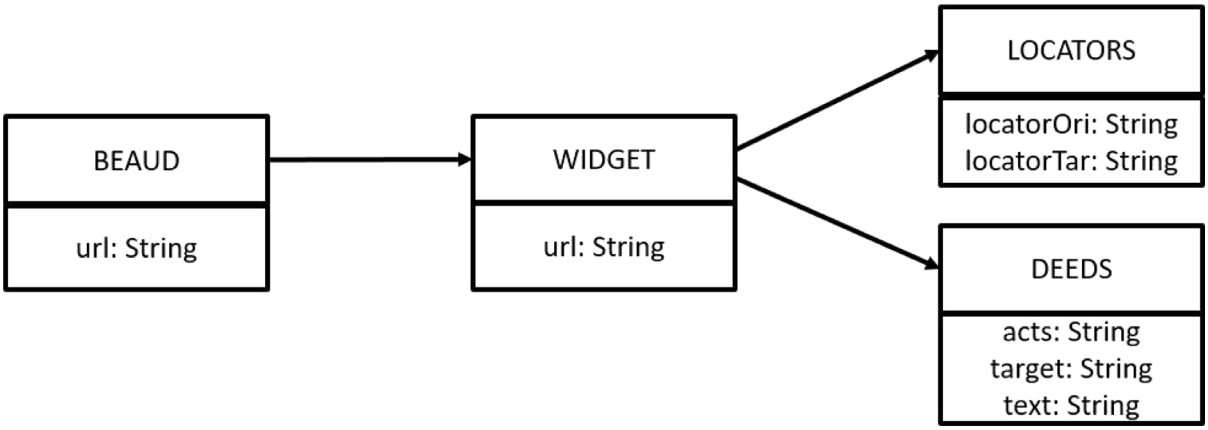

Fig. 1. Software architecture of BEAUD.

has disadvantages. For example, a high number of clicks is indicative of problems [6]. Spending an hour in front of the computer typing causes muscle fatigue in 66% of people [7]. One solution could be to copy and paste, but this action causes efficiency problems and errors when navigating through different tabs (tab-switching) [8]. Users spend 57.4% of their time tab-switching and find it difficult to locate the desired information [9]. The more tabs opened, the more difficult it is to find the desired information and in general, users have more than 8 tabs opened (66.9%) [10]. Moreover, when users are continually tab-switching to find the desired information, they easily lose concentration [11]. URL typing is another common action in which users make mistakes [12]. Finally, users have to scroll 76% of the sites to find the information they are looking for. In addition, 22% of the time, they scroll to the bottom of the site [13]. The fact of not finding the desired information makes users scroll continuously, increasing their frustration [6].

Therefore, a tool should reduce the number of user deeds. For this reason we have developed BEAUD (Browser Extension to Automate User Deeds), to reduce the number of user actions on the web saving time and effort.

2. Software description

2.1. Software architecture

The software architecture of BEAUD is shown in Fig. 1. This figure shows that all the architecture consist of four different components: BEAUD, widget, locators and deeds. BEAUD is developed in JavaScript. All the information needed for the correct execution is stored in the local-storage of the browser in JSON format.

BEAUD is the main component of the architecture. It stores the main information, the address of the web page on which the automation process should start running.

Widgets are those web elements from third party sites that will be inserted into the web page that is being adapted. It contains the URL of the website from which BEAUD will extract the desired web content. Widgets need two components of information: locators and deeds.

A Web locator is a mechanism for uniquely identifying an element on the Document Object Model (DOM) [14]. This component must collect the locators of both the page from which the widget is extracted and where it will be inserted in the adapted web page.

The deeds component collects chronologically the different actions that the user can do in his search for information: clicks, double clicks, copy, paste and write. In addition, of these actions, it must save the XPath of the element in which the action will be performed. Furthermore, if the action consists of writing, it must save the written text.

2.2. Software functionalities

BEAUD is a browser extension that works in Google Chrome and Microsoft Edge, the two most used web browsers at the moment[1]. These extensions, once installed in the browser, appear via the application icon on the top right. Some extensions are enacted every time a web page is visited. However, other applications are activated once the user clicks on the icon of the application the user wants to run. This is the case of BEAUD. Once the user clicks on the application icon, three different options are displayed: ”New”, ”Save” and ”Delete”. These three actions change BEAUD to its three different states: creation, automation and deletion. Each of these states will be detailed below. Table 1 illustrates these states with an illustrative example. Table 2 shows the number of necessary deeds to perform the information search with BEAUD and without it.

2.2.1. Creation

Once the user clicks on the ”New” option in the menu, BEAUD goes to the creation state. While the application is in this state, the background of the node on which the mouse is currently hovering will change color. In this way, the user, who wants to make the adaptation, knows at all times to which node is affecting what he is doing at that moment.

The first step is to enter the address of the web page from which the user wants to extract the information. To do this, the user must double-click and BEAUD will show the user under the node, which was selected, an input in which he must enter the web address from which the user wants to extract the information. Once entered, a new tab will be opened with the address entered.

The second step is to select the desired node. Similarly to the previous step, the background of the selected node will have a different color. In this step, the user will be able to do all the necessary deeds to select the desired node: scroll, click, copy&paste, etc. BEAUD is able to record these actions in order to reproduce them in the automation phase. What should not be done is to open a new tab because BEAUD will not collect this information. Once the desired node is visible, the user must hover over the node and when its background changes color, double-click. By this action, the extension saves the location of the node (locator) and closes the tab, returning to the page being adapted.

A special case in capturing user deeds is the automation of web searches based on a data that changes, for example, the title of a movie in IMDB. The user, before pressing the ”Go ahead!” button after entering the URL in the input, must copy the text from the adapting website. Following the IMDB example, the title of the movie. The user should proceed to search for information as he would do it manually, and BEAUD will pick up his actions.

[1] https://gs.statcounter.com/browser-market-share/desktop/worldwide

Table 1
Illustrative example.

Action	Description
Data based web search	For the example, we are going to use the DBLP[2] web page. This page collects and displays publications obtained mainly in the field of computer science. It classifies them into journals, conferences or books. However, it does not show data concerning the number of references obtained by these contributions. For this reason, if a user wants to know the number of references an author has, the user must copy the name of the researcher and paste it in the search bar of Google Scholar[3], for example. Google Scholar provides information about the total number of citations, the h- index and the i10 index. Using BEAUD, the goal is to insert this information into the DBLP web page. To carry out this process, the first step is to search for a researcher's profile in DBLP. For this example, the user has searched for his own profile. Once the user has visited Iñigo Aldalur's profile, he clicks on "New" in the BEAUD menu. By double-clicking on it, the user can enter the URL of Google Scholar. The next step is to select and copy author's name (see Fig. 2 top-left). Once clicked on "GO AHEAD!", the user must perform a normal search for information. In this case, paste the name copied previously in the search bar and click on the search button (see Fig. 2 top-right). The list of results shows the profile of Iñigo Aldalur and the user clicks on the corresponding result (see Fig. 2 bottom-left). Once the user has accessed the correct profile, he selects the box with all the cites of the author, and the user double-clicks. The tab closes, and the user will see the item transferred to the DBLP web page instead of the URL entry item.
Search for information without data	Let us think that the user wants information about a journal each time he visits the DBLP page to see if the impact of a particular journal has changed. For this example, the user will enter the impact factor of the journal Software Impacts[4]. Once the user double-clicks again on the DBLP web page, he must enter the web address of the journal (see Fig. 3 left). In this occasion, the user does not rely on data from the DBLP web page itself. The system captures the necessary interactions. For this example, the user only has to select the desired element and double click (see Fig. 3 right). The tab closes, and the user will see the element transferred to the DBLP web page instead of the URL input element as in the previous case. Once the adaptation process is finished, the user must click on "Save" in the BEAUD menu. In this way, the creation process is finished, enacting the automation process.
Parallel automation process	Once clicked on "Save" or each time the user visits the DBLP web page, the process is enacted. Once the process is started, the system enters messages in the places where it will replace the extracted information. These messages warn that if the content is not entered within a reasonable time by leaving the tab open, it is because there has been a failure. In the example, the content of Software Impacts is transferred before that of Google Scholar content, since the number of actions is lower. In the top of Fig. 4, it is possible to see the result of Iñigo Aldalur's profile in DBLP after the automation. Both external contents are surrounded by a frame to indicate that it is external content introduced with BEAUD. In case the user wants information about another person, he looks for the profile in DBLP and the system is activated again. In the bottom of Fig. 4, it is possible to see the information related to Urtzi Markiegi researcher. It can be seen that the values related to Google Scholar have been updated, showing the number of references of the current author.

The user can repeat the first and second steps as many times as necessary to adapt the web page to his desire or information needs.

To finish this process, the user must click on the "Save" option in the extension menu. By this action, the status changes to automation. Furthermore, the webpage is reloaded, and the automation process is enacted.

2.2.2. Automation

If the application is in this state, each time the user accesses the adapted web page, the system is automatically enacted, i.e. the user does not have to perform any action. Once the adapted web page is recognized, the system opens a tab for each of the elements to be inserted in the adaptation. The user will always see the web page that is being adapted, but the user will see that a certain number of tabs have been opened. BEAUD, in parallel, repeats each of the deeds that the user has carried out to obtain the desired node. Once the node is accessible, the node is picked up, the tab is closed and the node is inserted in the corresponding place. This process ends when each and every node has been inserted, and all open tabs have been closed. If, for some reason, one of the nodes is not accessible because some of the actions could not be performed, or the system does not find the desired node, the tab remains open. In this way, the user will be able to know which of the nodes has failed and in which step.

The execution of the special case differs from the others in the initial step. The system collects the text of the adapting website of which the user wants to do the search and will introduce this data into the search engine of the web page itself that will do the search. Afterward, the system will reproduce the captured user actions. When the system is in this state, the only option available in the application menu is "Delete". This action, once performed by the user, takes the system to the next state, deletion.

Table 2
Number of actions completed by users with and without BEAUD.

	Scroll	Click	Tab-switching	New tab	URL typing	Copy	Paste
Without BEAUD	0	1	4	2	2	1	1
With BEAUD	0	0	0	0	0	0	0

2.2.3. Deletion

Once the user selects this option from the extension menu, the system changes its status and the circle is completed. Once the action is done, the only option available in the extension menu will be "New".

During this process, the system removes all the information related to the adaptation in the system. This action is only available if the automation has been enacted, i.e. if the current page has an adaptation. This action reloads the web page, removing all the nodes inserted in the automation process.

3. Impact overview

Many WA related contributions have been published in recent years. [15] survey provides the description of the characteristics of WA tools and analyzes tools from 2007 to 2016. WA can provide help to end users in very diverse environments. Other more current contributions have tried to help people with disabilities [16], take- away food businesses [17] and recommendation systems (movies, restaurants, hotels...) [18]. Another new area has been mobile browsers. Mobile extensions have not been possible until a few years ago and for this reason, contributions in mobile WA are very recent. The fact of having smaller screens and that certain interactions are more complex, makes WA tools very useful to minimize interactions, decreasing errors and also help to obtain information much faster [19,20]. This is not all,

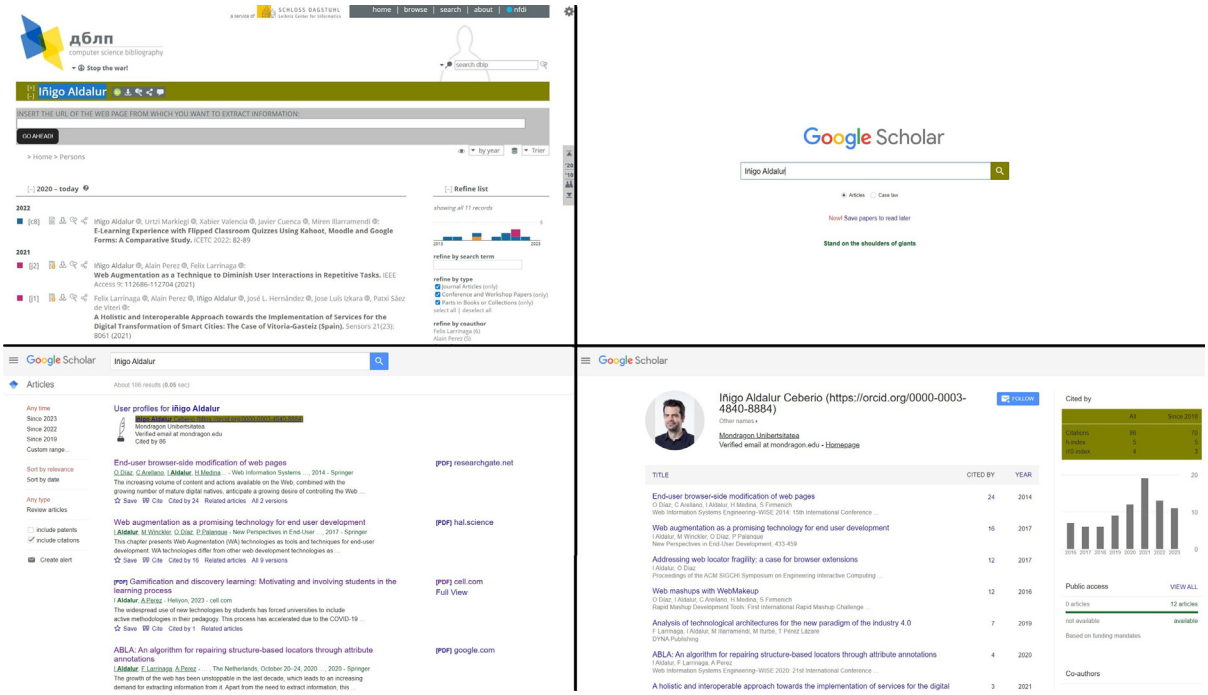

Fig. 2. Example that shows how to extract content from Google Scholar based on DBLP website data using BEAUD.

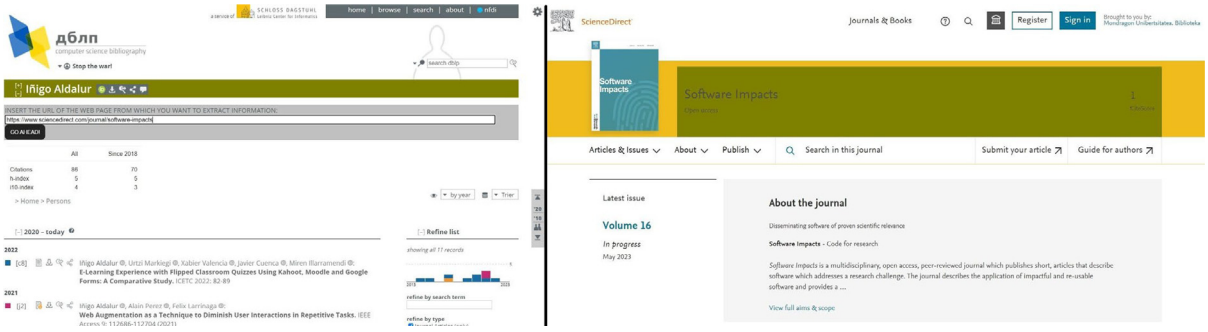

Fig. 3. Example that shows how to extract content from Software Impacts journal website using BEAUD.

WA has contributed to improve the interaction with semantic web pages [21] or Web of Things based applications (web pages that help end users with their Internet of Things devices) [22]. The impact of WA is important in different topics, but it is also important for end users to apply it to their most common web interactions [4,15,19,20].

On the other hand, the most widely used desktop browsers today are Google Chrome and Microsoft Edge. BEAUD is compatible with both browsers. This means that more than 77% of users can use this extension in their usual desktop browsers.

4. Limitations and future work

The main limitation of this tool is a problem that has been explained before, the update of a web page. These changes cause the mechanisms used to find the desired nodes (locators) to break down. Consequently, it is impossible to find the desired node and the automation stops. As future work, we want to add mechanisms that make the life of a locator longer [23–27]. In addition, we want to investigate and develop some mechanisms that further extend the life of the locators.

5. Conclusions

We have introduced BEAUD, a browser extension that uses WA techniques to automate any user's information search tasks. To complete the automation, users must enter the address of the page from which they want to extract the information. Once entered, users simply perform the usual steps to get the information they want and select the element that provides the information. Once the action is completed, BEAUD is able to perform all the defined actions automatically and in parallel. In this way, the number of interactions required is reduced to zero and user frustration is reduced as well.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was carried out by the Software and Systems Engineering research group of Mondragon Unibertsitatea (IT519-22), supported by the Department of Education, Universities and Research of the Basque Government, Spain.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.simpa.2023.100516.

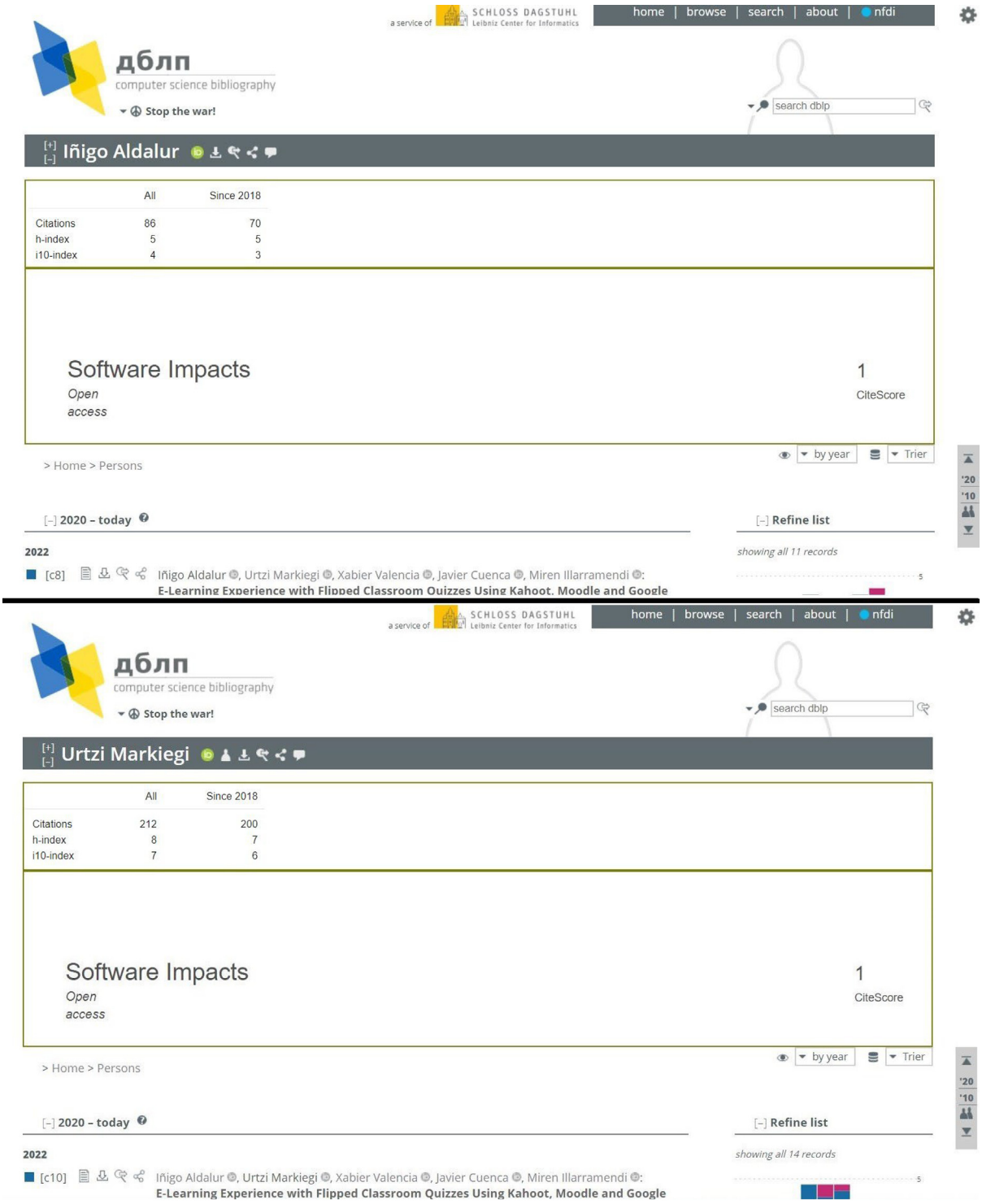

Fig. 4. Final results of the adaptation with BEAUD.

References

[1] M. Eirinaki, M. Vazirgiannis, Web mining for web personalization, ACM Trans. Internet Tech. (TOIT) 3 (1) (2003) 1–27.

[2] M.D. Mulvenna, S.S. Anand, A.G. Büchner, Personalization on the net using web mining: introduction, Commun. ACM 43 (8) (2000) 122–125, http://dx.doi.org/10.1145/345124.345165.

[3] N.O. Bouvin, Unifying strategies for web augmentation, in: HYPERTEXT '99, Proceedings of the 10th ACM Conference on Hypertext and Hypermedia: Returning to Our Diverse Roots, February (1999) 21-25, ACM, Darmstadt, Germany, 1999, pp. 91–100, http://dx.doi.org/10.1145/294469.294493.

[4] O. Dıaz, C. Arellano, M. Azanza, A language for end-user web augmentation:

Caring for producers and con- sumers alike, ACM Trans. Web 7 (2) (2013) 9:1–9:51, http://dx.doi.org/10.1145/2460383.2460388.

[5] T.Y. Lee, B.B. Bederson, Give the people what they want: studying end-user needs for enhancing the web, PeerJ Comput. Sci. 2 (2016) e91, http://dx.doi.org/10.7717/peerj-cs.91.

[6] A. Aula, R.M. Khan, Z. Guan, How does search behavior change as search becomes more difficult?, in: Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, Atlanta, Georgia, USA, April (2010) 10-15, ACM, 2010, pp. 35–44, http://dx.doi.org/10.1145/1753326.1753333.

[7] B. Callegari, M.M. de Resende, M. da Silva Filho, Hand rest and wrist support are effective in preventing fatigue during prolonged typing, J. Hand Therapy 31 (1) (2018) 42–51.

[8] K.T. Stolee, S.G. Elbaum, G. Rothermel, Revealing the copy and paste habits of end users, in: IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2009, Corvallis, OR, USA, 20-24 2009, Proceedings, IEEE Computer Society, 2009, pp. 59–66, http://dx.doi.org/10.1109/VLHCC.2009.5295296.

[9] J. Huang, R.W. White, Parallel browsing behavior on the web, in: HT'10, Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, Toronto, Ontario, Canada, June 2010 13-16, ACM, 2010, pp. 13–18, http://dx.doi.org/10.1145/1810617.1810622.

[10] J.C. Chang, N. Hahn, Y. Kim, J. Coupland, B. Breneisen, H.S. Kim, J. Hwong, A. Kittur, When the tab comes due: challenges in the cost structure of browser tab usage, in: CHI '21: CHI Conference on Human Factors in Computing Systems, Virtual Event/ Yokohama, Japan, May (2021) 8-13, ACM, 2021, pp. 148:1–148:15, http://dx.doi.org/10.1145/3411764.3445585.

[11] O. Dıaz, J.D. Sosa, S. Trujillo, Activity fragmentation in the web: empowering users to support their own webflows, in: 24th ACM Conference on Hypertext and Social Media (part of ECRC), HT '13, Paris, France - May (2013) 02-04, ACM, 2013, pp. 69–78, http://dx.doi.org/10.1145/2481492.2481500.

[12] R. Tahir, A. Raza, F. Ahmad, J. Kazi, F. Zaffar, C. Kanich, M. Caesar, It's all in the name: Why some urls are more vulnerable to typosquatting, in: 2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April (2018) 16-19, IEEE, 2018, pp. 2618–2626, http://dx.doi.org/10.1109/INFOCOM.2018.8486271.

[13] K. Athukorala, D. Glowacka, G. Jacucci, A. Oulasvirta, J. Vreeken, Is exploratory search different? A comparison of information search behavior for exploratory and lookup tasks, 67, (11) 2016, pp. 2635–2651, http://dx.doi.org/10.1002/asi.23617,

[14] F. Ricca, M. Leotta, A. Stocco, D. Clerissi, P. Tonella, Web testware evolution, in: 15th IEEE International Symposium on Web Systems Evolution, WSE 2013, Eindhoven, The Netherlands, September 27, 2013, IEEE Computer Society, 2013, pp. 39–44, http://dx.doi.org/10.1109/WSE.2013.6642415.

[15] I. Aldalur, M. Winckler, O. Dıaz, P.A. Palanque, Web augmentation as a promising technology for end user development, in: New Perspectives in End-User Development, Springer International Publishing, 2017, pp. 433–459, http://dx.doi.org/10.1007/978-3-319-60291-2_17.

[16] C González-Mora, I. Garrigós, S. Casteleyn, S. Firmenich, A web augmentation framework for accessibility based on voice interaction, in: Web Engineering - 20th International Conference, ICWE 2020, Helsinki, Finland, June (2020) 9-12, Proceedings, in: Lecture Notes in Computer Science, 12128, Springer, 2020, pp. 547–550, http://dx.doi.org/10.1007/978-3-030-50578-3_42.

[17] L. Goffe, S.S. Chivukula, A. Bowyer, S.J. Bowen, A.L. Toombs, C.M. Gray, Web augmentation for well- being: the human-centred design of a takeaway food ordering digital platform, Interact. Comput. 33 (4) (2021) 335–352, http://dx.doi.org/10.1093/iwc/iwac015.

[18] M. Wischenbart, S. Firmenich, G. Rossi, G. Bosetti, E. Kapsammer, Engaging end-user driven recom- mender systems: personalization through web augmentation, Multim. Tools Appl. 80 (5) (2021) 6785–6809, http://dx.doi.org/10.1007/s11042-020-09803-8.

[19] I. Aldalur, A. Perez, F. Larrinaga, MAWA: A browser extension for mobile web augmentation, in: Human- Computer Interaction - INTERACT 2021-18th IFIP TC 13 International Conference, Bari, Italy, August 30 - September 3, 2021, Proceedings, Part IV, in: Lecture Notes in Computer Science, 12935, Springer, 2021, pp. 221–242, http://dx.doi.org/10.1007/978-3-030-85610-6_14.

[20] G.A. Bosetti, S. Firmenich, S.E. Gordillo, G. Rossi, An approach for building mobile web applications through web augmentation, J. Web Eng. 16 (1 & 2) (2017) 75–102.

[21] C. Sottile, S. Firmenich, D. Torres, An end-user semantic web augmentation tool, in: End-User De- velopment - 7th International Symposium, IS-EUD 2019, Hatfield, UK, July (2019) 10-12, Proceed- ings, in: Lecture Notes in Computer Science, 11553, Springer, 2019, pp. 239–243, http://dx.doi.org/10.1007/978-3-030-24781-2_23.

[22] J. Lobo, S. Firmenich, G. Rossi, N. Defossé, M. Wimmer, Web of things augmentation, in: Proceedings of the Eighth International Workshop on the Web of Things, WoT 2017, Linz, Austria, October 22 2017, ACM, 2017, pp. 11–15, http://dx.doi.org/10.1145/3199919.3199923.

[23] I. Aldalur, F. Larrinaga, A. Perez, Abla: an algorithm for repairing structure-based locators through attribute annotations, in: Web Information Systems Engineering - WISE 2020-21st International Confer- ence, Amsterdam, The Netherlands, October (2020) 20-24, Proceedings, Part II, in: Lecture Notes in Computer Science, 12343, Springer, 2020, pp. 101–113, http://dx.doi.org/10.1007/978-3-030-62008-0_7.

[24] M. Biagiola, A. Stocco, F. Ricca, P. Tonella, Diversity-based web test generation, in: Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August (2019) 26-30, ACM, 2019, pp. 142–153, http://dx.doi.org/10.1145/3338906.3338970.

[25] M. Hammoudi, G. Rothermel, A. Stocco, WATERFALL: an incremental approach for repairing record- replay tests of web applications, in: Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November (2016) 13-18, ACM, 2016, pp. 751–762, http://dx.doi.org/10.1145/2950290.2950294.

[26] H. Kirinuki, H. Tanno, K. Natsukawa, COLOR: correct locator recommender for broken test scripts using various clues in web application, in: 26th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2019, Hangzhou, China, February (2019) 24-27, IEEE, 2019, pp. 310–320, http://dx.doi.org/10.1109/SANER.2019.8667976.

[27] A. Stocco, R. Yandrapally, A. Mesbah, Visual.web.test. repair, Visual web test repair, in: Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November (2018) 04-09, ACM, 2018, pp. 503–514, http://dx.doi.org/10.1145/3236024.3236063.