



A visual programming tool for mobile web augmentation

Iñigo Aldalur¹ · Alain Perez¹ · Felix Larrinaga¹ · Miren Illarramendi¹

Received: 27 October 2022 / Revised: 29 September 2023 / Accepted: 6 December 2023
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2024

Abstract

The use of mobile devices for web browsing has increased extraordinarily in recent years, becoming the main source of information. Unfortunately, developers cannot meet the needs of all users. As a result, users have been forced to adapt web content to their needs. Additionally, users face more costly interactions than on desktop computers due to the smaller screen size of cell phones. These interactions include URL typing, scrolling, tapping, branching and tab-switching. If the number of web resources is large, the activity becomes unpleasant. Furthermore, should the quantity of interactions be extensive, the web experience also becomes unpleasant. Moreover, all these interactions lead to higher costs in terms of phone time and battery life, as well as more costly navigation. Web augmentation enhances the user web experience by adding content from different resources, modifying the original layout, or reducing interactions during web searches, among other benefits. To reduce user interactions and satisfy their information necessities, we propose MAWA, an extension for Firefox mobile that allows users to adapt any web page to their needs through web augmentation techniques. To validate our proposal, an evaluation of the tool has been carried out. The article presents the benefits introduced by MAWA as a response to the inconveniences experienced by users when performing their activities in the mobile browser. The results obtained show that MAWA is able to reduce the number of interactions significantly, as well as to reduce battery consumption and the time needed to obtain the desired information. In addition, users perceive that navigation is less cluttered.

Keywords Human–computer interaction · End-user development · Web augmentation · Customization · Mobile browser

✉ Iñigo Aldalur
ialdalur@mondragon.edu

Alain Perez
aperez@mondragon.edu

Felix Larrinaga
flarrinaga@mondragon.edu

Miren Illarramendi
millarramendi@mondragon.edu

¹ Electronics and Computing Department, Mondragon Unibertsitatea, Loramendi 4, 20500 Mondragon, Gipuzkoa, Spain

1 Introduction

There has been a substantial rise in the usage of mobile phones in recent decades, leading to an increase in access to web applications [1]. The number of web users is increasing, as is the number of mobile users [2]. According to various reports, the demand for mobile data will skyrocket in the coming decade [3]. In 1995, Boehm et al. [4] predicted that there would be 55 million Web practitioners in the USA by 2005. Scaffidi et al. [5] predicted in 2005 that there would be 90 million end-users in the USA in 2012. If we extrapolate this amount globally, we can imagine that the real number today is uncountable. In reality, mobile traffic now accounts for more than half of all Internet traffic. End-user participation in website customization according to their needs and preferences has increased as a result of this growth [6]. End-users can customize web pages by changing the colors, relocating components, removing features, or adding new elements. In other words, they can tailor the functionality, content, and design to their needs. Since most web users around the world are simple customers with minimal programming expertise, the customization process is primarily performed by users with no programming skills. Web augmentation (WA) is a common technique for customizing existing third-party websites. Bouvin [7] originally coined the term WA in 1999 to describe a tool that “through integration with a Web browser, an HTTP proxy or a Web server adds content or controls not contained within the Web pages themselves to the effect of allowing structure to be added to the Web page directly or indirectly, or to navigate such structure. The purpose of such a tool is to help users organize, associate, or structure information found on the Web. This activity can be performed by a single user or in collaboration with others.” Augmentation is created by the web user, not the website owner. There are a multitude of techniques for implementing WA, but one of the most popular is to use visual programming techniques [8]. The use of graphics as a programming language is known as visual programming [9], which facilitates the development of new functionality in websites by end-users.

Lieberman et al. [10] define End-User Development (EUD) as “a set of methods, techniques and tools that allow users of software systems, acting as non-professional software developers, to create, modify or extend a software artifact.” End-users may begin with simple adaptation mechanisms and progress to more effective adaptation mechanisms without encountering insurmountable obstacles [11]. WA is an ideal EUD technique because it can be implemented gradually. WA can be used on any website and is carried out by end-users. Mobile users face the same disadvantages as desktop users. Initially, websites were designed primarily for desktop computers, as they were the only supported platform for accessing the Internet. However, technology has progressed to the point that mobile phones and tablets are now popular platforms for browsing websites using navigation browsers [12]. As a result of this natural evolution, web designers developed a routine of first designing for desktop computers with high screen resolutions. Subsequently, they would create a mobile-friendly version of the website, maintaining the same layout [13]. Since mobile versions are often modifications of desktop versions, problems can be exacerbated. WA can be used not only for desktop web versions but also for mobile web versions, which can be personalized by end-users. When browsing on a mobile device, users require resources and techniques that meet their needs. This paper looks into WA as a mechanism for end-users to customize web pages through a Firefox mobile browser extension for web adaptation.

To accomplish this work, we have employed the Design Science Research (DSR) methodology [14]. DSR has been defined as “research that invents a new purposeful artifact to address a generalized type of problem and evaluates its utility for solving problems of that type” [15]. The method consists of five steps. The first one is to identify and motivate the problem. “DSR

requires a profound understanding of the problem to be solved, the consequences to be alleviated, and the causes to be prevented” [16]. The next step in DSR is to define the objectives of the solution, which, in most cases, are accomplished through the development of some tool or application. The objectives always aim to alleviate some of the consequences of the problem or to prevent some of its causes [16]. Section 2 defines the problem, its causes, consequences and objectives. The third step of the DSR method is the design and development of the tool or application to be developed. The fourth step involves demonstrating through real cases that the developed tool meets the objectives. Section 3 shows steps 3 and 4 illustrating them with a real example. It describes MAWA¹ (Mobile Application for Web Augmentation), a browser extension that customizes mobile version web pages to adapt them to users’ requirements. Lastly, step 5 of the DSR consists of evaluating the developed tool by individuals not involved in the development. The objective of the evaluation is to demonstrate that the defined objectives are met. Section 4 presents the conducted evaluation. If any of the steps 3, 4, or 5 did not meet the defined objectives, the DSR proposes returning to step 2. The DSR is an iterative process that concludes when the defined objectives are satisfied. Section 5 discusses related work in order to give the reader an idea of what has been done in this area. Finally, Sect. 6 outlines features we would like to enhance in MAWA and concludes the paper.

2 Problem statement

2.1 The problem

At the beginning, all websites were designed primarily for desktops. Nevertheless, with the introduction of web browsing on mobile phones, this practice began to change. Browsing the web on cell phones was initially quite cumbersome, and developers started designing two different versions of the same website: one for desktop and another for mobile phones. However, technology is evolving and the concept of “Mobile First” is gaining widespread acceptance. Mobile First is a philosophy developed by Luke Wroblewski [17], which emphasizes prioritizing the mobile environment over the desktop.

Mobile First involves developing a website design starting from its most basic form, initially focusing on the small screen of a smartphone, and then iteratively enhancing it for higher resolution devices [17]. However, a significant number of web designers still begin their web development with a focus on desktop resolution and subsequently adapt it to the mobile paradigm [18]. Designing for desktop resolution first often leads to issues and disadvantages in the mobile web browser edition [3].

The process of adapting a web page to different screen resolutions is achieved through Responsive Web Design (RWD). RWD is a design and implementation technique that empowers web designers and developers to create websites with a flexible approach, allowing content and layout to adapt to various resolutions on different devices while maintaining a consistent web experience [19]. However, this adaptation is not without its costs. There is a computational overhead involved in adjusting the initial desktop version to suit the mobile browser configuration. Furthermore, this adaptation can result in usability issues, such as displaying unnecessary content, reordering important content, and, consequently, leading to cluttered navigation and increased scrolling to find the desired information, among other challenges [20]. These usability drawbacks induce a poor web experience for users.

¹ <https://addons.mozilla.org/es/firefox/addon/mawa/>.

RWD was originally conceived for adapting from desktop to mobile, even though there is a common sentiment that it should follow a “mobile-first design” approach [21]. RWD’s primary goal is to avoid the need to create two separate versions of the same website (desktop and mobile versions). Nevertheless, to cater to the diverse needs of all users, web developers would ideally create a version tailored to each user. However, this is practically unfeasible as each individual has distinct needs, interests, and preferences. A common technique for adapting content to user preferences is web personalization. “Web personalization is an automated process that identifies a user, collects his or her navigation patterns, analyzes known preferences of similar users, and estimates his or her specific preferences to tailor content for each user” [22]. However, personalization techniques may not always anticipate individual preferences accurately. The Web Augmentation (WA) technique is better suited to meet users’ specific needs.

2.2 Causes

The reasons why it is unviable to develop a web version for each user and the RWD implementation are:

- **No cost-effective:** Web development companies are not interested in developing a personal software or different versions for each user. The reason is that the more clients a company has, the more versions it would need to implement. The cost associated with producing these personalized web versions would be prohibitive for most companies.
- **Not easy to foresee:** It is impossible to design a web solution that can provide information for every conceivable situation or include personalized content for each individual user [23]. While there are applications that automate and personalize web interactions [24, 25], these tools are insufficient to guarantee the fulfillment of user objectives when updates to the relevant context cannot be fully predicted at the design stage [26].
- **User effort:** Since web developers are generally not inclined to create different web versions to cater to individual needs, end-users are compelled to undertake this task themselves. Most end-users lack programming knowledge, and the effort required to make web adaptations using programming languages is substantial. One of the primary objectives of end-user augmentation tools is to reduce the effort required by end-users to adapt websites, thus minimizing the gap in execution [27].
- **Small screen:** Various research studies have investigated the issues stemming from small screen sizes and their effects. For instance, [28] concluded that small screen sizes may be a contributing factor to users providing lower-quality data when using mobile phones compared to desktop computer users when responding to questionnaires. Meanwhile, [29] compared error rates based on screen size and found that larger device sizes lead to better authentication rates. [30] determined that screen size impacts both readability and comprehension. Finally, [31] deduced that individuals who view only a small portion of a website often lack a sense of overall context, and they may feel lost on a large page, being forced to remember the locations of desired web elements and scroll repeatedly to access them.

2.3 Consequences

The consequences of the invalidity to develop a web version for each user and the RWD implementation are:

- **Scroll:** [32] conducted a study in which they determined that people used the scroll bar on 76% of pages, and 22% scrolled to the bottom regardless of the length of the page. A frequent action in browsing is to scroll up and down a page without reading the content [33]. This might be a signal of frustration and a lack of confidence [33]. In fact, Lagun et al. [34] concluded that users are less satisfied when they frequently scroll to find irrelevant information during their navigation. Given that, “the current scrolling method for a mobile device is both time-consuming and fatigue-prone” [35], it is mandatory to reduce scrolling on mobile devices.
- **Taps:** Tapping is the action of clicking on a web element on the mobile screen and it requires only a very small scale of device movement (of the order of 1 mm) and can be performed with the finger or palm of the hand [36]. A high number of taps by a user can be indicative of problems [33]. For example, it has been observed that users tend to make more taps than necessary during search tasks and this is due to the tendency of users to make unnecessary taps [37]. The more taps a user makes, the poorer the interaction becomes. Additionally, taps on mobile devices produce errors in the interaction depending on how the user holds the device [38].
- **Tab-switching:** Taking into account that users switch tabs at least 57.4% of the time [39], it is extremely important to reduce this behavior. Constant tab-switching is a common way to access information from websites in different tabs. Nevertheless, if users have to switch tabs frequently to locate previously found information, they do not actually aspire to gather information from these sites [40]. Another study concluded that misplacing a few tabs in the wrong window or tab can have adverse consequences, such as missing important information buried in the wrong task [41]. When juggling multiple activities, users are required to memorize and understand the information contained in each tab [42].
- **Branching:** Branching is defined as the act of initiating a new tab (or window), which allows people to pleasantly navigate multiple websites concurrently [43]. However, this concurrent navigation comes with its own set of challenges. According to [41] concluded that if the number of open tabs increases, users find it more difficult to find relevant information or identify which tabs to focus on. They also found that on average, users have 8 tabs open and 66.9% of the subjects encounter one or more issues every week due to the number of open tabs. Managing a large number of tabs in mobile browsing often involves suspending, resuming, and interleaving actions [44].
- **URL typing:** Several studies that have analyzed the errors associated with typing URLs [45]. “Users prefer the former way because typing on screen-limited devices is difficult and chances of entering the wrong URL are high” [46]. It is important to remark that the difficulty of typing a specific domain name or the probability of making a particular types of mistakes is not purely random [45].
- **Battery consumption:** Battery consumption is an important concern in mobile device usage. Despite improvements in battery life expectancy, users have increased their device usage, particularly for activities such as social networking [47]. The activities that consume the most energy include social networking, gaming and searching on the internet [48]. Different mobile phone components consume different amounts of energy [49], with the screen being the most significant contributor, accounting for approximately 35% of the device’s total power consumption [50]. Reducing screen-on time and optimizing internet searches are essential to reduce battery consumption.
- **Time:** Time is closely related to the previous topic, as users aim to obtain their desired information as quickly as possible to minimize device usage. However, there are factors beyond the control of WA. For example, web page load times depend on various factors,

including search loops, file writing, and page loading via HTTPS, which can take between 7 and 13 s [51]. Additionally, the location of the server has an impact on web page load times in different countries. In one study, the average load time for European countries was 4.72 s, while for the USA, it was 7.64 s. It's worth noting that the bottleneck for modern browsers is no longer the network connection [52].

- **Produce cluttered navigation:** The process of adapting website structure from the desktop version to the mobile version involves rearranging the layout and content to fit the screen resolution [12]. This adaptation process introduces various disadvantages, such as hindering the legibility of content on the device [53]. This drawback results in increased user interactions on mobile devices compared to desktop users, consequently extending the time required to complete tasks [54]. Furthermore, it leads to more input errors compared to desktop navigation [55].
- **Show unnecessary content and hide important content:** The main characteristic of RWD is the relocation of the content displayed in the desktop version, which may involve hiding some content while keeping other elements visible in the mobile version [19]. Unfortunately, the displayed content might prove irrelevant to users, and worse, if essential content is concealed, users must engage with the website to locate the necessary information. In some cases, the simplest course of action for users is to scroll to find the desired information. Considering that all mobile web users want to access the desired information quickly and with minimal mental effort [56], it is imperative to present content effectively.
- **Computational cost reordering content:** RWD techniques modify the appearance of the web pages to adapt to the display limitations of a device. Reordering the layout implies a computational cost due to rendering [57]. The challenge arises when page layout issues occur during rendering for different display port sizes. The primary problem lies in the difficulty of validating all possible layouts [58].

Our initial objective is to develop a tool that alleviates all the consequences outlined in this section. This is an ambitious objective, especially considering that the tool should be user-friendly for end-users as well.

3 Augmenting mobile websites

Nowadays, it is more common to use mobile phone browsers than desktop browsers.² Therefore, it is of utmost importance to adapt mobile versions of websites to meet the needs of users, aiming to minimize or eliminate scrolling, tab-switching, branching, and reduce battery consumption, among other factors. We will illustrate the use of MAWA with an example based on the BBC website.

MAWA is a Firefox mobile browser extension for WA designed to remove, move and add content from any website through a visual programming environment. MAWA abstracts technical details into a visual DSL that facilitates end-user involvement. The primary goal of this extension is to address the causes and consequences presented in Sect. 2. The actions of moving, removing, and adding content are the key steps to mitigate these issues. The new contribution presented in this paper differs significantly from what was presented at the INTERACT2021 conference [59]. At the conference, a very early version of MAWA was introduced, allowing only the deletion and movement of content on a web page.

² <https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet>.

Today, only a few mobile browsers allow you to install extensions, with Firefox and Safari being the most popular options. While Firefox's usage stands at 0.6%,³ it is available on both Android and iOS. As seen in related work, mobile extensions have primarily been developed for Firefox, rather than other browsers, due to this reason.

3.1 First steps

In general, the number of interactions that a user must carry out in a mobile phone browser is greater than in desktop computer browsers [60]. For example, due to the smaller screen size, mobile users often need to scroll to find desired information, even when it's located at the top of the website. Responsive Web Design compacts web menus, requiring users to tap on the menu button before selecting an option. Therefore, it is crucial to simplify interactions as much as possible when developing extensions. In desktop computer browsers, extensions are typically displayed in the top left corner. By simply tapping on the extension and selecting a menu option, the extension is activated. However, mobile browser extensions are typically hidden, necessitating a sequence of actions: selecting the browser menu, then complements, followed by the specific extension, and finally, the menu option. This results in four interactions, compared to two for desktop browsers. Considering this, it becomes evident that a simple interaction to access the menu is necessary. Double-tapping is not a commonly used interaction on mobile phones. Consequently, our proposal is to activate the menu by double-tapping on the screen, revealing different options that users can access (see Fig. 1). Deactivated options are displayed in gray, while each active button has its own distinct color. Users can only tap the colored buttons. Tapping the "Cancel" button allows users to continue normal navigation. The "New" button initiates the WA process. The "Mineit" button can be used to copy a web node from the currently viewed website. Once the WA process is complete, an activated "Save" button would save the adaptation, and the 'Remove' button would delete the saved adaptation.

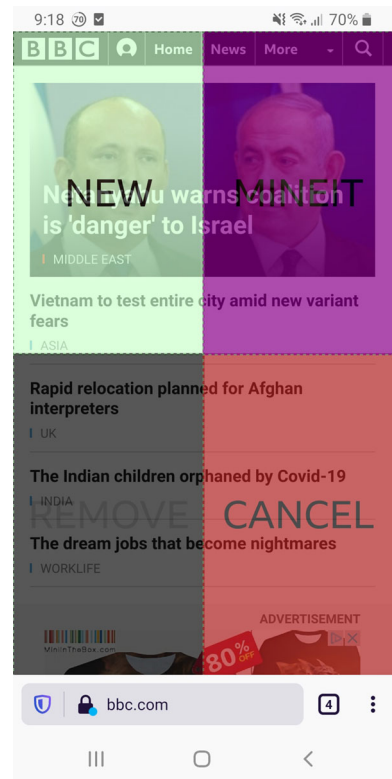
However, browser extension menus are not the only drawback; node selection is not a basic action in mobile web browsers. Web programmers are familiar with the DOM structure and how it is organized. Nonetheless, end-users do not have knowledge about the organization of the DOM. WA extensions are designed for users with no technical knowledge, and they must be able to complete any action permitted by the extension.

To enable end-user node selection, we have included four buttons: one at the top, one at the bottom, and one on each side of the selected node (see Fig. 2). Any node can be selected by tapping on it, but some nodes cannot be selected simply by tapping due to their peculiarities. Added arrows allow users to select relative nodes. For example, in the left image of Fig. 2, a weather image node is selected on the BBC website. The left arrow permits the selection of the previous node, and the right arrow is used to select the next sibling. Tapping on the arrow button at the top of the selected node selects the parent node (right image in Fig. 2). In the right image of Fig. 2, an arrow button at the bottom is included. Tapping it selects the first child node. If the selected node does not have any children, this button is hidden (as shown in the left image of Fig. 2). Similarly, if the selected node lacks a previous or any following siblings, those buttons are also hidden. The top button is only hidden when the body node is selected.

This mechanism allows the selection of any interactive node indirectly. Furthermore, nodes larger than the mobile screen resolution can be selected by tapping the top arrow button, and even complete tables can be selected using the same process. Thanks to this mechanism, any

³ <https://gs.statcounter.com/browser-market-share/mobile/worldwide>.

Fig. 1 Mobile extension menu for web augmentation



node can be selected, enabling the accomplishment of WA tasks, including the selection of difficult-to-select nodes in the augmentation process.

3.2 Move

The first action implemented in MAWA is the movement of web nodes from one position to another. The main goal of moving content is to eliminate the need for scrolling. Each website has its own structure designed by the web designer, which may not always align with all user preferences. For instance, consider a user who frequently reads the “World in Pictures” section on the BBC website, which is located at the bottom. The user has to scroll through the entire website to access this section, even though they would prefer it at the top. MAWA allows users to easily move this node to the top by using a drag-and-drop mechanism. When a node is dragged, the background color of the potential drop location turns purple, providing the user with visual feedback on the new node’s placement. Once the node is dropped, it is inserted into its new location and hidden from the original position. The reason for hiding the element is to ensure that the copy reflects the most up-to-date content.

In Fig. 3, on the left (A), you can see a node selected by the user within the “World in Pictures” section of the BBC website. In the middle (B), the figure illustrates how the user is relocating the node to the top of the website, with a purple background indicating the insertion point. On the right (C) of Fig. 3, the node has been successfully inserted at the top of the website. This example demonstrates how MAWA effectively prevents the need for

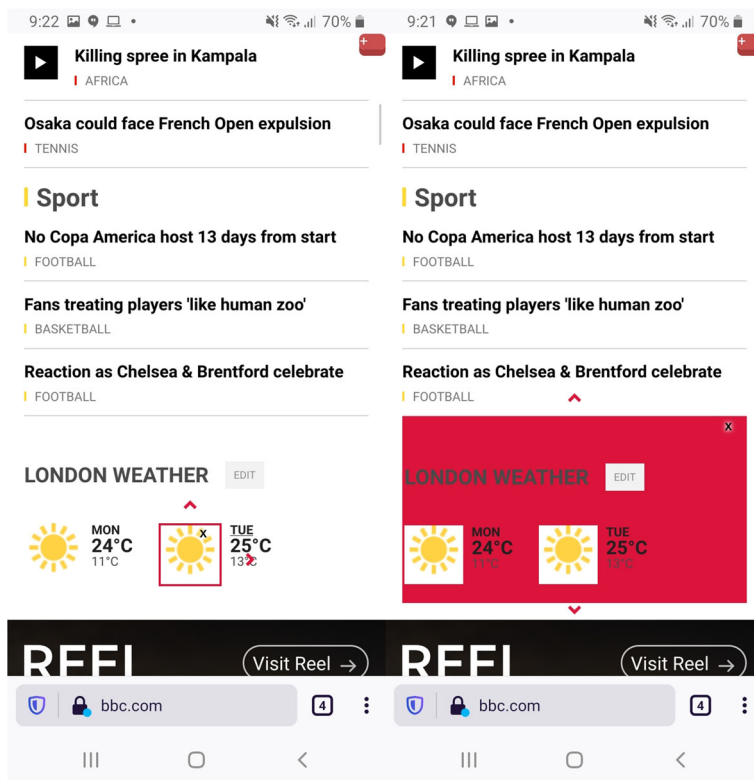


Fig. 2 Node selection including relatives buttons: leaf node (left), leaf node father (right)

scrolling when essential or important information is located at the bottom of a webpage and, consequently, also avoiding cluttered navigation.

3.3 Remove

The second action implemented in MAWA is the removal of unnecessary nodes for the user. The main goal of this action is to prevent the user encountering and consuming content that the web designer might have considered useful but is irrelevant to the user. Websites often present information that is not relevant to the user's needs. While such information may not be bothersome on desktop computer screens, it can be frustrating when viewed on small cell phone screens, as reported in studies [54, 55].

Continuing with the BBC example, imagine that the user has no interest in sports, making this information useless to them. The best solution would be to remove this content from the site. To do so, the user can click the 'X' button located in the upper right corner of the selected node (Fig. 4 left). Once clicked, the node is removed from the web page (Fig. 4 right). Subsequently, every time the user reloads or revisits the website, this node will remain hidden, simplifying the website's content. Thus, MAWA not only eliminates cluttered navigation but also ensures that the user avoids consuming unnecessary content.

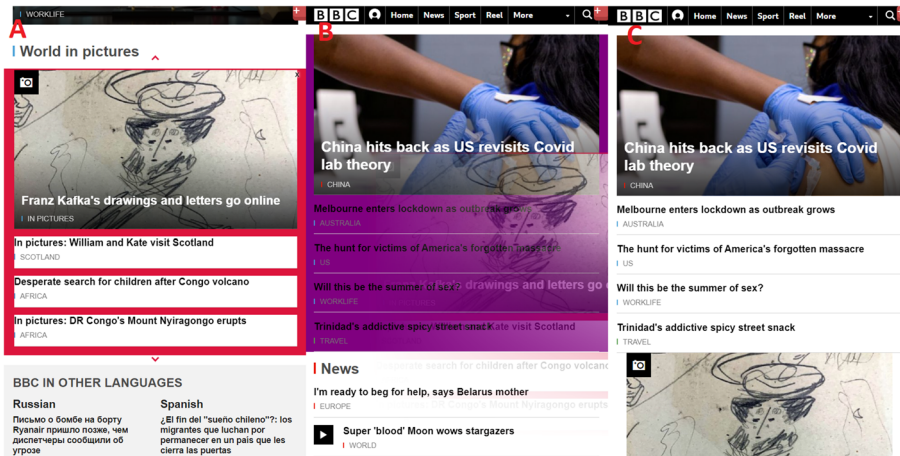


Fig. 3 Moving a node from the bottom to the top of the BBC website

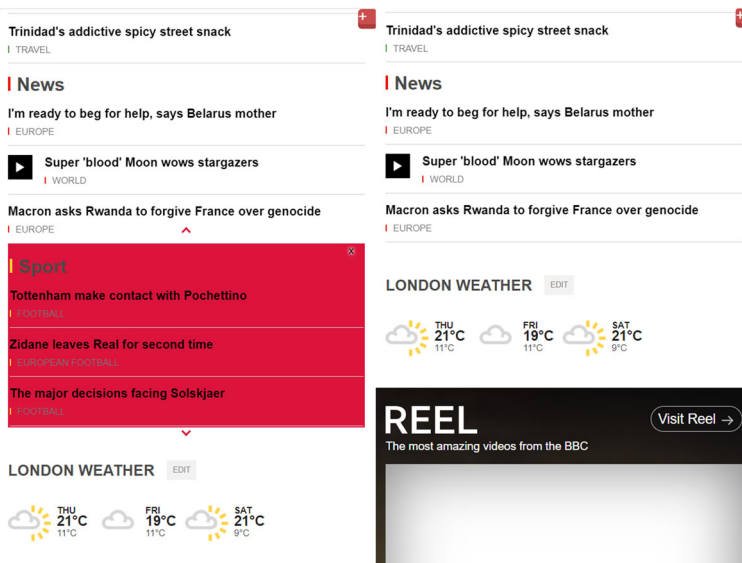


Fig. 4 Removing a node from the BBC website

3.4 Inserting foreign content

Not only does node relocation and removal offer a solution to mitigate the problems and consequences discussed in Sect. 2, but adding content from different resources is also a method to alleviate these drawbacks. We can distinguish between two types of foreign content: constant nodes whose content does not require updating and those that need to be kept up-to-date.

The creation process is the same for both. When a user visits a website and double-taps on the screen, the extension menu displays the “MineIT” option (see Fig. 1). Once the user clicks on it, the creation process begins. Similar to the editing process for moving and removing



Fig. 5 Mining a web node

web nodes, when the user taps on a web element, the background color of the node changes. The difference is that this node contains a button in the middle with the word “COPY” (see Fig. 5 left). If the user clicks on this button, MAWA prompts the user to insert a name for this copied element to complete the creation process. Clicking “Accept” reloads the page and saves the element, while clicking “Cancel” concludes the creation process without saving the element (see Fig. 5 right).

The main goal of inserting content from different websites is to avoid or reduce the number of taps, tab-switching, branching and URL typing. Most users visit a second website to complete their user experience [24]. If the consumed content is automatically inserted into the initial website, it eliminates branching and, consequently, URL typing. Furthermore, the user does not need to change tabs because everything is within the same tab, reducing the need for tab-switching. This reduction in tab-related actions significantly decreases the number of taps required by the user.

3.4.1 Updating content

Some nodes’ content must be upgraded because their nature is dynamic, and the content is updated by the website owner frequently. When the user wants to insert a node, he must tap on the “+” button that is always located on the top right of the screen. Figure 6 shows the list of copied components after tapping the “+” button. The list can be hidden by clicking on the “−” button in the top right corner. To insert an element, the user must tap on the element he wants to insert and then tap on a web element to insert it after this element. Figure 7 left shows how the user has inserted the program of the BBC channel from a TV guide.⁴ The BBC channel schedule is updated every hour and the user would like to access the latest content whenever they visit the site. This new element includes two different buttons. On the right, the “X” button can be used to remove the element. On the left, the “Constant/Update” button allows the user to specify whether this element should be updated or remain constant. Figure 7 on the right shows the element with a green background, indicating that this element will be updated every time the user visits the website. Please note that the content of this button changes from “Constant” to “Update” depending on its state.

⁴ <https://www.tvguide.co.uk/>.

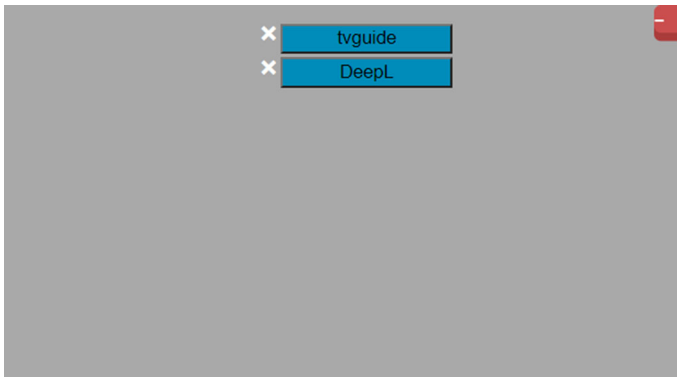


Fig. 6 List of nodes copied from different websites

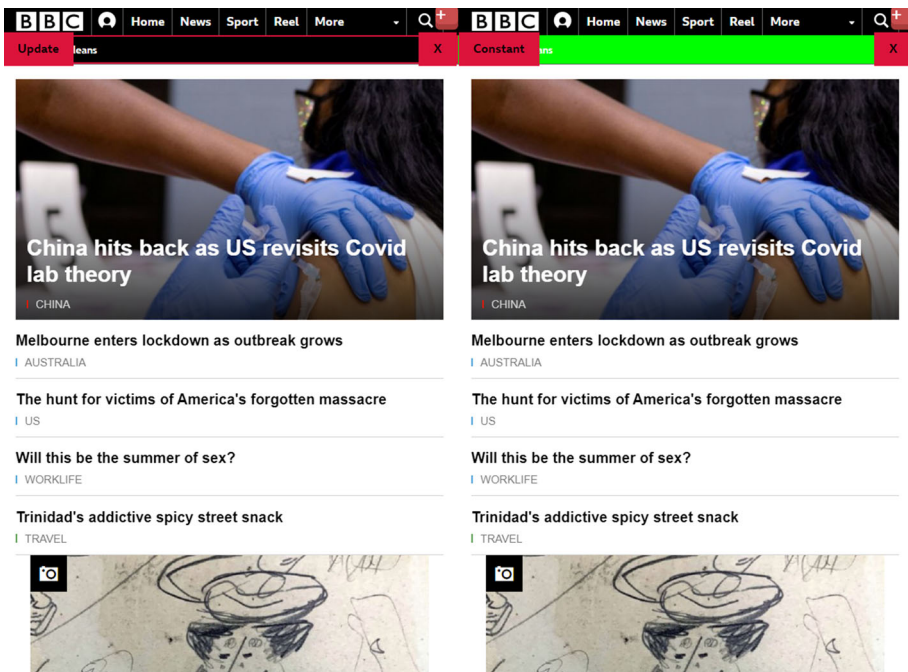
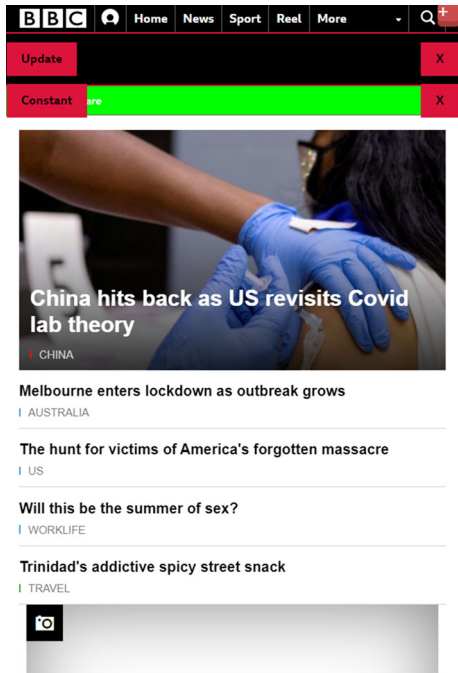


Fig. 7 Updating content from foreign element

3.4.2 Constant content

Some nodes will remain constant, meaning that the information they contain is crucial for the user, and it should not be updated even if updates are available on the original site. To illustrate this with the BBC example, consider a non-English speaker who wants to translate a news article into their language. Figure 8 shows how the user has inserted DeepL content at the top of the website, making it easier to access this translation service. This content must always remain the same, independently of the day.

Fig. 8 Constant content from foreign element



3.5 General improvements

MAWA provides three different actions (move, remove, and add) that have been designed to reduce or eliminate the causes and consequences of the problem presented in Sect. 2. These actions were developed to directly mitigate these drawbacks, including scrolling, hiding unnecessary content, tapping, tab-switching, branching, and URL typing. Moreover, MAWA has been designed to address other disadvantages as well. Cluttered navigation is mitigated through node removal and content relocation, allowing users to adapt site content to their needs. As a comprehensive solution, MAWA attempts to minimize the impacts of the underlying causes. It now enables the possibility of having different versions of a website for various users, even without the direct involvement of the website owner. The visual programming interface is designed to simplify the adaptation process for end-users, including those with no programming knowledge, reducing the effort required to adapt any website. However, certain limitations remain. It is impossible to change the small screen resolution of mobile phones, and the computational cost associated with reordering content cannot be completely eliminated, since MAWA does not remove RWD operations. Additionally, there is some computational cost involved in reordering the added, moved, and removed nodes. To validate these statements, an evaluation has been carried out.

3.6 Data model

The WA process with MAWA is divided into two different phases: the editing phase and the execution phase. The data model has been designed to consider the information that the system must store in both phases. The editing phase records the actions performed by

the end-user, while the execution phase retrieves this previously saved information, which MAWA uses to execute the WA process. We leverage the browser's structure to store this information. The data model is illustrated in Fig. 9.

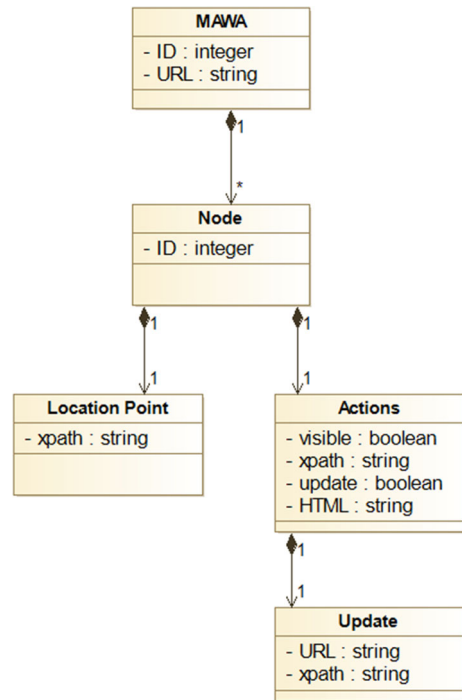
Every augmentation performed by MAWA is assigned a unique ID for registration purposes. Additionally, it saves the URL at which the augmentation will be executed. When the end-user introduces a new URL or reloads a web page, MAWA's engine checks whether this URL has been associated with an augmentation. If the URL is stored in the system, it initiates the process to execute the augmentation. Otherwise, the system waits until another web page is loaded. It's worth noting that MAWA allows for multiple adaptations for the same domain, but it is only triggered when the current URL matches the one defined during the design phase.

Each augmentation consists of one or more nodes that will be modified or inserted during the editing process. All nodes are identified by a unique ID. The information for each node includes its location point on the page and the actions that MAWA will perform when the augmentation is enacted.

The location point is defined by a locator, specifically an Xpath. A web locator can be described as a mechanism for uniquely identifying an element in the Document Object Model (DOM) [61]. This locator provides information about the node's location on the website, which is relevant for nodes that will be moved or removed. Nodes inserted from external websites do not require location point information to complete their action.

There are three different actions, and MAWA must save the appropriate information to complete all of them. The Xpath is used to locate the position in which the moved node and the inserted nodes will be inserted. This field is empty when a node is removed because the location point already provides this information. The 'visible' attribute is used for removed elements and moved ones. When an element is moved, a copy is inserted into the new location, and the original one is hidden. If an update to the content of this node has been made, the user will see the updated content rather than the old one. This is why the HTML attribute is empty for move and remove actions and is used to store the HTML code of foreign nodes only. The 'update' attribute indicates whether the HTML must be updated or not. If this attribute is set to true, the 'Update's URL' field specifies the website from which the new HTML content must be retrieved, using the information provided by the Xpath locator.

In the previous subsections, we mainly explained the editing process facilitated by the actions implemented in MAWA. However, this editing process serves as the preparatory step for the execution process. The goal of the execution process is to replicate the user's creation steps one by one. Hence, MAWA stores information about each modification made by the user during the editing process, following the described data model. The automation process begins by identifying the first modified element, and then reproduces the user's actions by deleting and relocating all nodes in the same order as the user followed during the editing steps. However, for every foreign element that needs updating, it is upgraded in parallel. The execution time depends on the time needed to update the slowest website, and can also be influenced by the connection speed. At the end, the layout will be the same as the one the user had at the conclusion of the editing process. However, the main drawbacks of the automation process are related to website updates. These updates can cause MAWA to be unable to find the modifiable nodes, leading to automation failures. These failures can occur in three different cases. The first case is when the node to be deleted or moved has been deleted from the website or relocated, making it impossible for the locator (mechanism used to locate nodes within a website) to find the node. In this case, if the node is to be deleted, the user doesn't perceive any error. However, if the node is to be moved, the user may notice that an important element is missing. The second case is when the node selected as a reference

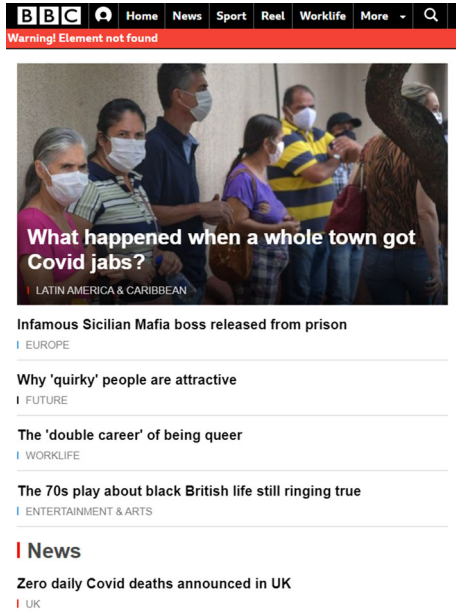
Fig. 9 MAWA data model

for relocating the node has been removed or relocated. In this scenario, the system is unable to relocate the node, and it appears as though the node has simply been removed from the website. The third case occurs when the website from which MAWA must extract updated content has been updated, and this update affects the element. In this case, the system inserts a failure message instead of the element itself (see Fig. 10). If any of these cases occur, the end-user can update the adaptation by double-tapping on the screen and selecting 'update'.

4 Evaluation

We conducted two types of user tests to evaluate MAWA. The first type was a quantitative evaluation, in which we compared user actions in real tasks from their daily lives with the actions they performed while carrying out the same activity with MAWA. The measured actions were the quantitative ones previously discussed in the causes and consequences described in Sect. 2. The second type of evaluation was qualitative, involving subjects who performed a guided activity using MAWA, and we collected qualitative data from their experiences. To gather this data, we employed a standard questionnaire known as PSSUQ (Post-Study System Usability Questionnaire), along with general questions based on the causes and consequences explained in Sect. 2. PSSUQ [62] assesses user satisfaction following participation in an instrument evaluation.

Fig. 10 Failure updating a foreign element



4.1 Quantitative evaluation

This section of the evaluation appraises the causes and consequences of Sect. 2 that can be quantitatively evaluated including scrolling, tapping, tab-switching, branching, URL typing, battery consumption, execution time and rendering.

4.1.1 Battery consumption

AccuBattery⁵ application has been used to measure the energy consumption. This application can measure energy consumption by tracking individual applications, starting from when the mobile phone is fully charged or powered on. Each execution was conducted with the mobile phone fully charged.

Figure 11 left shows the battery consumption by Firefox applications executing the BBC example with MAWA, the same example described in Sect. 3. On the contrary, Fig. 11 right shows the battery consumption of Firefox while performing the same actions without MAWA. The battery consumption with MAWA was 1.5mAh, whereas the battery consumption without MAWA was 12.1mAh. This indicates that MAWA reduces battery consumption by a factor of 8 compared to performing the same content consumption without MAWA, resulting in an 87.6% reduction in battery consumption.

Furthermore, an additional experiment was conducted involving 10 use cases, comparing MAWA's performance with and without MAWA. The battery consumption with MAWA was 302.6mAh, while without MAWA it was 24.8mAh, resulting in a reduction of 91.8%.

⁵ <https://play.google.com/store/apps/details?id=com.digibites.accubattery>.

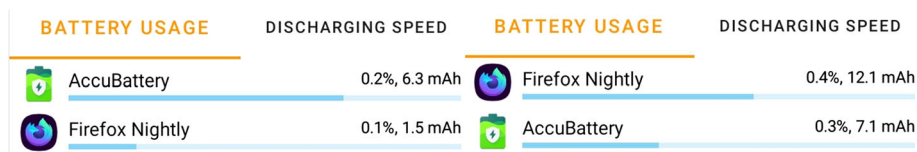


Fig. 11 (Left) Battery consumption with MAWA, (right) battery consumption without MAWA

4.1.2 Computational cost reordering content

The Firefox Developer Tool Inspector has been used to measure the time needed to render the website layout. This tool allows recording the execution time of various aspects of a web page loading such as the rendering, loading or scripting. Each execution is not exactly the same and the time needed to load is different. Since each execution may vary slightly, it has been executed the BBC example 20 times with and without MAWA in order to obtain a more precise rendering execution time. The average rendering time without MAWA has been 815.7 ms. The average rendering time without MAWA was 815.7 ms, while the average rendering time with MAWA was 891.95 ms. The difference between the two, with and without MAWA, was 76.25 ms, representing a 9.18% increase. We can consider this difference to be sufficiently small and irrelevant from the user's perspective.

4.1.3 Diminishing user interactions and consumption

The aim of this study was to compare information consumption on the web with and without MAWA and measure the reduction in interactions and resource consumption.

Research Method

Settings. The study was conducted at Mondragon University (Arrasate - Mondragon, Spain). All participants used their own mobile phones, on which they had installed Firefox Browser (Nightly for Developers). This browser permits the installation of Firefox extensions.

Procedure. At the beginning, subjects were individually evaluated, and they were unaware of the purpose of the activity. They only knew that they would be monitored while performing a task they frequently do, which they chose at that moment. The next day, they were asked to use MAWA to complete the same task, and they were monitored again. The task involved adding content to a national TV guide website from filmaffinity, rottentomatoes, and a TV guide website. Additionally, they had to remove channels they did not like and relocate their favorite channels to the top. Afterward, participants were given the opportunity to adapt another website they frequently visit. On this day, each subject was provided with an explanation of how to use MAWA. The reason for conducting the second part on the following day was that users tend to forget the results from the previous day, and this allowed them enough time to recall their actions.

Subjects. Five use cases were performed by 20 end-users, who did not have any programming knowledge skills. Each subject works in different economic sectors; research, medicine, sales, plumbing, driver, etc.

Instrument. Each subject completed the activity independently while a reviewer recorded every action performed during the activity. We measured scrolling, taps, tab-switching, URL typing, branching, time, battery consumption, and computational cost for reordering content. These measurements align with the quantitative metrics identified in Sect. 2.

Results

The evaluation results are summarized in Table 1. All the issues in both tables refer to the consequences described in Sect. 2. During the evaluation, a reviewer recorded every single action performed by the subjects both with and without MAWA. The top section of Table 1 displays, in the first five columns, the number of each action performed by the subjects in their activity without using MAWA. Battery consumption and rendering cost were calculated independently after the user completed the activity, following the same method explained in Sects. 4.1.1 and 4.1.2. The last column calculates the time needed to complete the activity. Conversely, the bottom section of Table 1 shows the actions performed by the subjects in the same activity using MAWA. Battery consumption, rendering cost, and time were calculated in the same manner. The first group of users compared the same product on different websites, such as book prices on Amazon and Powell's. The second group searched for information related to an upcoming journey. The third group looked for information about movies before making a selection. The fourth group was interested in sports and sought information about their favorite sports. The last group of users searched for gossip and yellow press-related information.

During the activity, the reviewer documented every single action accomplished by the subjects. Tab-switching was noted when the user changed tabs, taps when the user touched the screen to interact with a web element, scrolling when they scrolled to find information not initially visible on the website. Branching involved opening a new tab, and URL typing occurred when the user entered a URL in the tab. These actions are related because when the user opens a new tab, they typically introduce a new URL. In some cases, the user entered the URL in an already open tab, but in most cases, a new tab was opened. Finally, the time measured the duration of the activity in seconds. Battery consumption and rendering cost were calculated using the same method explained in Sects. 4.1.1 and 4.1.2.

Comparing Table 1 top and bottom (i.e., without MAWA and with MAWA), it is evident that the number of actions and values in all columns (with the exception of the rendering cost) has significantly decreased for all use cases when using MAWA. The first five columns represent the actions performed:

- Use Case 1 (UC1): The number of actions without MAWA was 49.1, and with MAWA, it reduced to 1.4, marking a reduction of 97.15% in the number of actions.
- Use Case 2 (UC2): Actions without MAWA numbered 53.9, while with MAWA, they reduced to 2.8, resulting in a 94.8% reduction.
- Use Case 3 (UC3): Without MAWA, there were 33.6 actions, but with MAWA, they dropped to 1.6, signifying a 95.23% reduction.
- Use Case 4 (UC4): Actions without MAWA were 34.7, and with MAWA, they decreased to 1.6, equating to a 95.39% reduction.
- Use Case 5 (UC5): Without MAWA, 43.5 actions were performed, while with MAWA, they reduced to 2.5, reflecting a 94.25% reduction.

To sum up, across all use cases, the number of actions was reduced by 95.39% (from 214.8 to 9.9) when using MAWA. Regarding battery consumption, the total consumption was 142.2 mAh without MAWA and 10.3 mAh with MAWA, signifying a reduction of 92.75% in energy consumption. The rendering cost was 9.18% higher with MAWA compared to without. Lastly, the time to complete the task was 84.83% lower when using MAWA (from 822.222 s without MAWA to 124.704 s with MAWA).

The quantitative evaluation confirms that some of the consequences of the problem described in Sect. 2 are solved or mitigated by MAWA. The number of user interactions decreased a 95.39%, the execution time was 84.83% and battery consumption was 92.75%.

Table 1 Number of actions completed by users in mobile phones without MAWA and with MAWA

	Scroll	Tap	Tab-switching	Branching	URL typing	Battery consumption	Rendering cost	Time
<i>Without MAWA</i>								
UC1	11.9	22.6	8.5	3.3	2.8	25.2	1356	162,334
UC2	15.1	20.5	9.7	4.6	4.0	33.6	1129	201,658
UC3	8.4	15.3	6.3	2.2	1.4	22.8	1533	122,419
UC4	9.3	11.8	7.7	3.1	2.8	32.1	969	149,287
UC5	11.2	17.3	8.1	3.8	3.1	28.5	1027	186,524
<i>With MAWA</i>								
UC1	1.2	0.2	0	0	0	1.7	1518	20,112
UC2	2.4	0.4	0	0	0	2.6	1234	32,913
UC3	1.4	0.2	0	0	0	1.6	1621	22,025
UC4	1.6	0	0	0	0	2.4	1102	26,357
UC5	2.5	0	0	0	0	2.0	1091	23,297

Nevertheless, rendering cost is 9.18% more elevated. Although the increase in rendering makes battery consumption higher, the drastic reduction in execution time makes battery consumption much lower. Taking into account the benefits obtained in the other issues, the increase in rendering time to approximately 100 milliseconds is acceptable.

Sections 4.1.1 and 4.1.2 affirm that the results calculated for the BBC example explained in the article are very similar to those calculated in the use cases evaluation.

In addition, the weights of each use case necessary to carry out the augmentation have been measured. This information is stored in JSON format in the browser following the model explained above (see Fig. 9). The weights vary according to the HTML that is saved and needed in the augmentations, the element with the highest storage. For UC1, the weight is 24KB, for UC2 36KB, for UC3 26KB, for UC4 31 KB and for UC5 33KB.

4.2 Qualitative evaluation

This section of the evaluation appraises all the causes and consequences of Sect. 2.

Research Method

Settings. The study was conducted at Mondragon University (Arrasate - Mondragon, Spain). All participants used their own mobile phones due to Covid19 measures, on which they had installed the Firefox Nightly app.

Procedure. At the outset of the evaluation, participants were informed of the study's purpose and provided with a brief description of it. Thereupon, a video featuring a MAWA instance was presented to illustrate the application's main functionality. The example involved adding content to a national TV guide website from Filmaffinity, Rottentomatoes, and a TV guide website. Additionally, participants had to remove channels they disliked and move their favorite channels to the top. Following this, participants were asked to adapt another website they frequently visited. Finally, participants were directed to an online Google Forms questionnaire.

Subjects. Twenty-four people participated in the evaluation, with 75% of the participants being male. The participants hailed from Arrasate-Mondragon and nearby towns. None of them possessed technical knowledge, as the evaluation aimed to test MAWA with end-users. All participants were between 18 and 39 years old. 70.8% of participants had installed at least one extension in their mobile phone browser. 54.2% of participants visited more than 5 web pages on their mobile phones every day, and only one subject did not use their phone to browse the internet every day. 54.2% of the subjects had more than 6 tabs open during each session on their mobile phones, and no one had only one tab open. 70.8% of the participants spent more than 30 min on the internet with their phones every day.

Instrument. To gather user experiences in the evaluation, a questionnaire was used. The questionnaire consisted of sections on background, usability (PSSUQ questionnaire), and general questions related to the causes and consequences of Sect. 2. General questions were measured using various items with a 7-point Likert scale (1=completely disagree, 7=completely agree).

Data Analysis. Descriptive statistics were used to characterize the sample and evaluate the participants' experiences using MAWA.

4.2.1 PSSUQ

The PSSUQ questionnaire is widely used to evaluate the usability of a tool. PSSUQ consists of 16 questions designed to measure usability. This usability scale was utilized in this evaluation

to gauge the overall usability assessment from the user's perspective regarding MAWA. The questionnaire involves averaging responses across three sub-scales: System Quality (the average of items 1–6), Information Quality (the average of items 7–12), and Interface Quality (the average of items 13–16) [63]. It uses a 7-point Likert scale (1=completely disagree, 7=completely agree) and includes an additional option for “NA” (Not Applicable). The results of the PSSUQ questions are reported in Table 2 and summarized in Fig. 12.

With regard to the System Quality questions, all average values fall between 2 and 3, with the exception of the fifth question, which is below 2. It is possible to conclude that participants perceive MAWA as a high-quality tool. They find it easy to use (Q1 AVG 2.16) and straightforward (Q2 AVG 2.21). The lowest values reported were 5 and 4, respectively, and only a few participants gave such high ratings. According to their feedback, users were able to complete activities quickly (Q3 AVG 2.58), and they felt comfortable using the extension (Q4 AVG 2.71). Finally, users believe that MAWA is easy to learn (Q5 AVG 1.71) and that they will be productive using the tool (Q6 AVG 2.66).

As for the Information Quality, some values also fall between 2 and 3, except for question 7, which is slightly higher than 3, and questions 9 and 10, which are below 2. The results indicate that MAWA's Information Quality is excellent. Question 7 inquires about the appropriate messaging of errors. MAWA provides information about element extraction failures. However, as the lowest-rated usability aspect, this feature should be improved (Q7, AVG 3.04). The ability to recover previous versions after making a mistake using the tool is well-received by participants (Q8 AVG 2.46). Participants found the information provided about the extension to be appropriate (Q9 AVG 1.96), and it was easy to find the information they needed (Q10 AVG 1.96). Additionally, participants believed that the information provided to complete the activity was correct (Q11 AVG 2.37). Finally, they found the on-screen information to be suitable (Q12 AVG 2.33).

As far as the Interface Quality is concerned, all results fall between 2 and 3. Hence, it is possible to conclude that participants found the interface to be outstanding. In general, they found the interface pleasant (Q13 AVG 2.66), and they enjoyed using it (Q14 AVG 2.83). Participants believed that MAWA provides all the functionalities expected in a tool of this kind (Q15 AVG 2.95). Finally, all participants expressed satisfaction with the system (Q16 AVG 2.08).

Standard deviation measures the amount of variation or dispersion in a set of values. A lower value indicates that most values are close to the average value. For most questions, the standard deviation is close to 1, indicating that participants' opinions are quite similar. The lowest standard deviation value is for Q10, likely because most participants found it easy to locate the information needed in the provided document. One reason could be that all participants' opinions were below 3 (the highest possible value on the questionnaire). The second-lowest standard deviation is for Q5, suggesting that, for most participants, MAWA is easy to use. Most values fall between 1 and 2, with a median of 1.5. The highest standard deviations are found in Q7 and Q13, the two questions with a maximum value of 7. This is because opinions on these issues varied more among participants.

4.2.2 General questions

Before the general questions, in the background section of the questionnaire, subjects were asked about some of the causes and consequences of Sect. 2. 83.3% of the subject must always scroll before finding something of interest on their mobile phones. They consider that in 54.4% of the websites they visit, there is unnecessary content and in the 50% of the websites interesting content is hidden due to RWD. They consider that in 58.3% of the websites, they

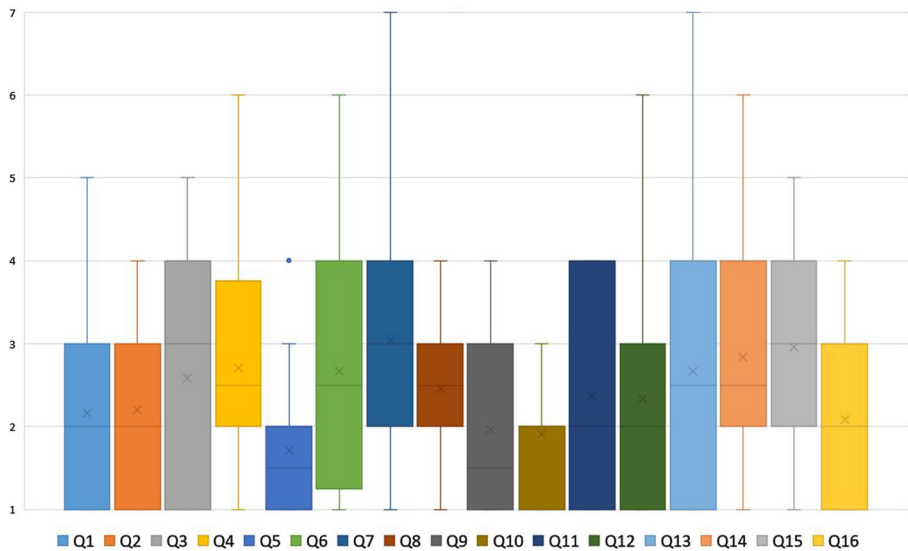


Fig. 12 PSSUQ scores

Table 2 PSSUQ results

	AVG	Med	SD	Max	Min
Q1	2.16	2	1.2	5	1
Q2	2.21	2	1.18	4	1
Q3	2.58	3	1.38	5	1
Q4	2.71	2.5	1.3	6	1
Q5	1.71	1.5	0.85	4	1
Q6	2.66	2.5	1.4	6	1
Q7	3.04	3	1.57	7	1
Q8	2.46	2.5	1.02	4	1
Q9	1.96	1.5	1.16	4	1
Q10	1.92	2	0.72	3	1
Q11	2.37	2	1.21	4	1
Q12	2.33	2	1.37	6	1
Q13	2.66	2.5	1.55	7	1
Q14	2.83	2.5	1.46	6	1
Q15	2.95	3	1.37	5	1
Q16	2.08	2	1.1	4	1

have problems finding their desired information. 62.5% of them claim that mobile browsing is really complex. The battery life is less than 1 day for 66.7% of the subjects. In general, they affirm that they perform a large number of clicks, they have a large number of tabs open, and continuously switch tabs in their mobile browser operations. They also consider that they make mistakes when writing by hand and remembering website URLs. Finally, they assert that mobile browsing is more complex than desktop computer browsing and that the small screen size of mobile phones is a drawback for browsing.

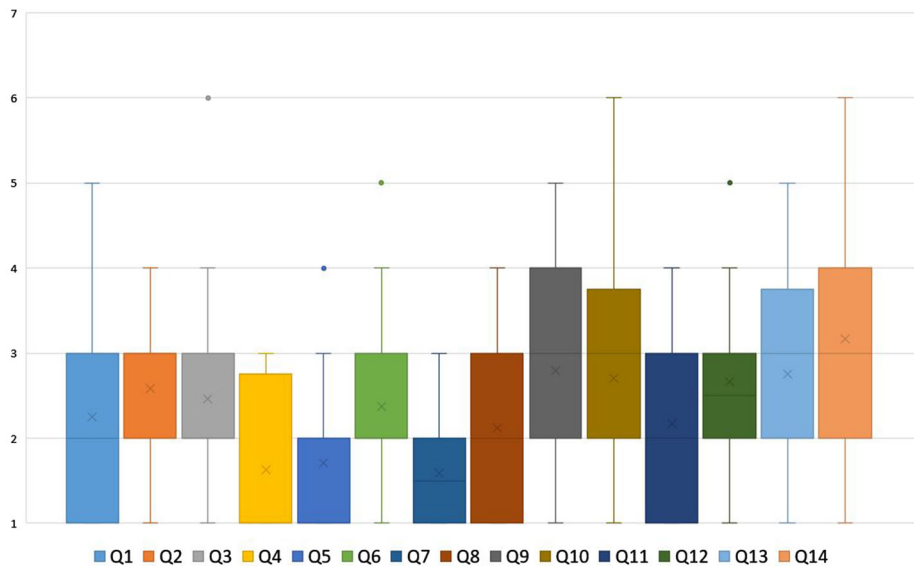


Fig. 13 Question scores

Table 3 Question results

	AVG	Med	SD	Max	Min
Q1	2.25	2	1.15	5	1
Q2	2.58	3	1.02	4	1
Q3	2.45	2	1.21	6	1
Q4	1.62	1	0.88	3	1
Q5	1.71	1	0.95	4	1
Q6	2.37	2	1.06	5	1
Q7	1.58	1.5	0.65	3	1
Q8	2.12	2	1.03	4	1
Q9	2.79	3	1.17	5	1
Q10	2.71	3	1.33	6	1
Q11	2.16	2	1.13	4	1
Q12	2.66	2.5	1.05	5	1
Q13	2.75	3	1.26	5	1
Q14	3.16	3	1.43	6	1

General question results are reported in Table 3 and summarized in Fig. 13. In this questionnaire, a Likert scale is used from 1 to 7. These questions evaluate subjects' feelings about the influence of MAWA in mitigating or eliminating the causes and the consequences of the problem described in Sect. 2. The more causes and consequences are solved, the greater the influence of MAWA in solving the problem. Questions 1 to 4 concern the causes, and questions 5 and 14 pertain to the consequences described in Sect. 2.

For this questionnaire, a 7-point Likert scale has been used (1=completely disagree, 7=completely agree). The questions have been written to determine whether MAWA solves or helps to diminish their impact on mobile browsing.

- Q1 No cost-effective: In the subjects' opinion, it is not economically feasible to develop a different version of the same website for every user. However, they believe that MAWA would help in having different versions of the same website. The average rating for this question is 2.25, with a median of 2, a maximum score of 5, and a minimum of 1.
- Q2 Not easy to foresee: Participants' opinion averages 2.58, with a median of 3, a maximum rating of 4, and a minimum of 1 related to this topic. They consider that MAWA facilitates adding content to any website. Web developers do not need to study the profiles of the users to develop different versions; the users themselves will develop their own version with MAWA.
- Q3 User effort: The subjects' opinion is that MAWA helps personalize any website to the end-users' needs in an effortless manner. The average rating for this question is 2.45, with a median of 2, a maximum score of 6, and a minimum of 1.
- Q4 Small screen: Subjects believe that MAWA helps with browsing despite the small screen size. It is impossible to increase the screen size with a browser extension. However, the overall functionality of MAWA alleviates this drawback. The general opinion score is 1.62, with a median of 1, a maximum value of 3, and a minimum of 1.
- Q5 Scroll: Subjects consider that the MAWA extension avoids scrolling. By moving content from the bottom to the top of the website, it eliminates this common action. Their opinion is 1.71, with a median of 1, a maximum of 4, and a minimum of 1.
- Q6 Taps: Adding content from different resources reduces the number of taps the user carries out in their browsing activity. Participants in the evaluation agree with this statement, as they believe, with an AVG value of 2.37, that this statement is true. The median is 2, the maximum value is 5, and the minimum is 1.
- Q7 Tab-switching: This is a very common action in web activities because usually, the needed information is divided among different websites. If the user desires to obtain complete information, they must continuously switch tabs. Participants claim that MAWA avoids tab-switching. Adding the necessary information from different websites to a single one avoids this action because all the information is on the same site. The subjects' opinion on this statement is 1.58, with a median of 1.5, a maximum value of 3, and a minimum of 1.
- Q8 Branching: This action is related to tab-switching. Before switching tabs, the user must open a new tab. For the same reason, adding content from different websites into a single one avoids opening new tabs. Subjects also agree with this statement because their overall opinion is 2.12, with a median of 2. The maximum value is 4, and the minimum is 1.
- Q9 URL typing: If the user does not need to open a new tab to obtain information, they won't have to type new website addresses. Due to this, mistakes will not occur in URL typing. Despite the fact that the subjects' opinion is one of the highest in this topic, participants also agree with this statement (AVG 2.79, median 3, minimum 1, and maximum value 5).
- Q10 Battery consumption: Taking into account that screen brightness deeply affects battery life expectancy and overall screen use [64], reducing the time a user has a lit screen is mandatory. In the quantitative evaluation, it has been demonstrated that MAWA executes web actions faster; hence, screen use will be reduced. Participants affirm this statement, believing that MAWA reduces battery consumption (AVG 2.71, median 3, minimum 1, and maximum 6).
- Q11 Time: The faster an activity is performed, the better it is for users. They will have more time for additional activities. Users claim that MAWA reduces the time needed to

complete a web activity, with an overall opinion of 2.16, a median of 2, a minimum value of 1, and a maximum of 4.

- Q12 Produce cluttered navigation: Reorganizing the website is one of the main objectives of MAWA. In the subjects' opinion, MAWA reduces cluttered navigation (AVG 2.66, median 2.5, minimum value 1, and maximum 5).
- Q13 Show unnecessary content and hide important content: RWD hides some content that is visible in the desktop computer version. In addition, it maintains some unnecessary information for some users. Removing this content is a feature of MAWA. This might be the reason why subjects think that the extension helps to remove unnecessary content and show important content (AVG 2.75, median 3, minimum 1 and maximum value 5).
- Q14 Computational cost reordering content: RWD has a computational cost to reorganize all the content. Moving, removing, and adding content also have computational costs, as demonstrated in the quantitative evaluation. This statement has received the highest rating from the subjects (AVG 3.16, median 3, minimum 1, and maximum 6). This affirms that users believe that MAWA does not significantly increase the computational cost.

The standard deviation of all values in the questionnaire is close to the value of 1. This means that all subjects' opinions are similar. Question 7, which is related to tab-switching, has the lowest value. With a median of 1.5, this indicates that most of the responses were either 1 or 2 on this issue. The highest standard deviation value is found in question 14, which relates to mobile computational cost. Considering that the computational cost is increased with the use of MAWA, subjects' opinions might be more diverse. Some may think that MAWA reduces the computational cost, while others believe the opposite.

4.3 Discussion

Our main objective with MAWA was to develop a tool that would alleviate all the causes and consequences of the problem described in Sect. 2 as well as improve its usability. To check if we have met the objective, different evaluations have been carried out, some quantitative and some qualitative. Among the quantitative evaluations, 3 different tests have been performed. The first one was to check if MAWA helps reduce battery consumption. The test shows that MAWA greatly reduces battery consumption. The weakness of this evaluation is that it has only been tested with one example, the one described in the article, and it is possible that in all cases, the reduction of consumption might vary. However, it is proven that MAWA reduces battery consumption by alleviating this consequence. The second test consisted of checking the computational cost of reordering content. In all WA tools, the computational cost of reordering content is increased. The evaluation shows an increase of 9.3%, a total of 76.25ms, which is an increase not perceptible by the user. As in the previous test, this evaluation has been carried out with a single example but no significant changes would be presented in other examples since this reordering is done in a single step in all cases. Finally, the third test consisted of evaluating 5 different use cases and testing the number of actions with and without MAWA. This evaluation shows that the number of actions by the user is drastically reduced with the use of MAWA. Moreover, the number of tab-switching, branching and URL typing is reduced to zero and the number of taps is almost zero. Scrolling is the only action that has been used, but its reduction is significant. The total reduction of actions was 95.39% which shows that MAWA is useful for reducing user actions. This test also shows a significant time reduction (84.83%) when the user is consuming the desired content. This third test reinforces the first two tests that had only been tested with one case. These new cases corroborate battery reduction and computational cost reordering content.

The average reduction of battery in all the examples run is 92.75%, a value even higher than the one shown in the first test. The increase in the computational cost of reordering content was 9.18%, a value very similar to the second test of the quantitative evaluation. Although five use cases are not a large number of cases to evaluate, all cases yield very similar results considering the experiments are of different nature and characteristics. We consider that this number of use cases is representative and have not conducted further tests. The entire quantitative evaluation shows that MAWA alleviates the consequences of Sect. 2, except for cluttered navigation, showing unnecessary content, and hiding important content, which are tested in the qualitative evaluation.

The qualitative evaluation consists of two distinct tests. One aims to demonstrate that end-users perceive that MAWA alleviates the consequences of Sect. 2. The second qualitative test determines whether MAWA is usable for end-users. The same end-user test has been used to evaluate both. The PSSUQ survey was used to assess usability. More than 75% of the subjects rated all questions below 4, the mean value of the scale. Additionally, the median of all questions is 3 or below. This shows that users perceive MAWA as usable for end-users. On the other hand, users have been asked whether they perceive that MAWA alleviates all consequences and prevents the causes. The results collected show that their perception is that MAWA alleviates the consequences and serves to prevent the causes. The disadvantage of this evaluation is that end-users would not know as well as a developer whether the causes of cost-effectiveness, ease of foreseeability, user effort, and small screen would be prevented by MAWA. Although we wanted to know their opinion, it is not an objective of MAWA. However, their perception of the consequences not evaluated in the quantitative part (producing cluttered navigation, showing unnecessary content, and hiding important content) is to be taken into account. We have considered end-users' opinions, and these are positive. These users endorse the improvements that MAWA produces in these aspects and refute the consequences evaluated in the quantitative part.

One of the significant challenges in technology design and deployment is ensuring accessibility and inclusivity across diverse user groups. Our experiments primarily involved participants with a strong educational background, which allowed us to assess MAWA's capabilities and limitations within this context. However, we recognize that a more comprehensive evaluation should consider users who may have less formal education or varying degrees of digital literacy. To address this concern, future research endeavors should focus on conducting user testing with a broader and more diverse range of participants, including those with varying educational backgrounds. This will provide insights into how MAWA can effectively serve a wider audience. Moreover, adapting the user interface and interaction design of MAWA to be more inclusive and user-friendly for individuals with varying levels of digital literacy will be a priority. This might involve simplifying workflows, providing clear instructions, and enhancing onboarding experiences.

5 Related work

5.1 Layout adaptation

As defined by Bouvin [7], WA modifications can be applied to website content, layout or behavior. Some tools have been designed to impact only the layout. For example, CrowdAdapt [65] focuses on enabling end-users to adapt the interface according to their desires or needs if they find the current web page design inadequate. CrowdAdapt provides a visual design

environment that augments web pages loaded in the browser, allowing users to customize the layout. The changes are deployed on a server in the form of a new layout template, which is automatically downloaded by the browser. These modifications can be immediately shared with other users. UIWear [66] is a client-side tool designed to help end-users easily create layout modifications without requiring access to application source code. UIWear automatically extracts the elements that compose the layout, after which the user can customize the interface based on their preferences by simply dragging and dropping elements. W3touch [67] is built using jQuery for the client and PHP with MySQL on the server. It has been designed to leave control with web designers while allowing end-users to improve layout visualization. It enables zooming in on the most crucial elements of the website to improve their readability. This modification is accomplished by tapping and selecting modifiable elements on the device screen. PageTailor [68] is a plugin that runs on Windows Mobile PDAs and is based on the Firefox browser. The tool employs a technique for selecting objects that involves dragging a line over the object to choose the corresponding web node. PageTailor allows traditional websites to be customized adjusting by the size of important web elements while reducing the size of irrelevant information. PageTailor is not used on websites that implement Responsive Web Design. Since Collapse-to-Zoom [69] only allows expanding and collapsing web objects, it is a much simpler tool than PageTailor. It can also be used to make changes to conventional websites on PDAs. The node is extended when the user taps over an element and drags the pen diagonally to the end. The element is collapsed if the movement is toward the bottom.

5.2 Layout and content adaptation

Nonetheless, some other tools have been designed to have an impact not only on the layout but also on the content. For example, SAM [70] is a client-side framework that permits the adaptation of website menus by end-users. It allows users to change the style and order of menus, personalizing them for better accessibility to the most visited sections. SAM primarily focuses on modifying website menus, whereas MAWA allows reordering not only menus but the entire web page. It's important to note that MAWA does not perform style modifications, such as changing colors or fonts. In contrast, W3Touch, PageTailor, and Collapse-to-Zoom modify the size of web page elements, either making them larger or smaller, which can be seen as a form of removal. PageTailor and Collapse-to-Zoom are designed for PDAs and traditional web pages that do not use Responsive Web Design. MAWA, on the other hand, does not allow changing the size of web elements but does enable their removal when necessary. UIWear also allows reorganizing the content of a web page, similar to MAWA. The key difference is that UIWear automatically selects the nodes, and the user decides the order of the nodes, effectively allowing users to remove nodes (similar to deletion). In contrast, CrowdAdapt is a server-side tool, distinguishing it from MAWA. This application also permits reorganizing content, similar to MAWA. However, CrowdAdapt goes further by allowing users to change the size and font of elements and even transform text into two columns, actions not supported by MAWA.

5.3 Content adaptation

Modifying or adding content from different resources is a common functionality in WA tools. Trusty et al. [71] developed ALOE, a Firefox browser extension that enhances web pages by converting all selected English words into their respective language translations. Sticklet

[72], another Firefox browser extension, allows users to add content from external websites to enhance the content of the target web page. To facilitate this process for users with little or no programming knowledge, a Domain Specific Language (DSL) has been created. This DSL helps users create their “sticks,” and a debugger is available to assist in correcting mistakes. Creators can also share their augmentations with the community. ENIA [73] presents a tool that proposes a data acquisition scheme capable of capturing web interface user interactions. These interactions can be replicated automatically without human intervention, reducing task completion time. Similarly, Excore [24] is a program that adds content to a website based on a series of web searches. It detects a keyword on a specific website, performs a web search on a different website, and then inserts the requested content on the target site. Users define the automation through a visual programmable Chrome browser extension. Taking a step forward in web adaptations, Chudnoskyy et al. [74] developed Omelette. It provides end-users with a composition environment comprising various elements that can be placed on the canvas, synchronized with each other, and shared with other users.

ALOE differs from MAWA in that it focuses exclusively on translating content into other languages. ENIA and Excore are Capture and Replay tools that automatically repeat all user actions to obtain the required information. In contrast, MAWA accesses the required content in a single step. Omelette allows users to insert predefined elements and their own content, while Sticklet also allows the insertion of content from other pages, similar to MAWA. However, the significant difference is that MAWA is a Visual Programming tool, whereas Sticklet is a DSL that requires users to learn its language for use. Sticklet provides a guide to help users without computer knowledge create their own “sticks,” which are embedded as notes on a board that users can access by clicking on them.

5.4 Content and behavior adaptation

Not only is content adapted by WA tools, but new behavior is also added. CrowdMock [75] is a Firefox extension. It differentiates between developers and consumers. The first ones create the augmentations and share their creations with the consumers. Additionally, developers can collaboratively create augmentations. Besides adding content from different resources, it allows the addition of new behavior to the website that facilitates navigation between different websites. MoWA [6] is a mobile browser extension that adds details to existing Web Applications that the original web page lacks. It adjusts the website content based on data collected from mobile sensors (GPS, microphone, gyroscope, light...). WebMakeup [76] is a Chrome browser extension for visual programming. Different web nodes can be copied by clicking on them with the mouse. Users can drag-and-drop previously copied nodes from a “Piggybank” into a website. By inserting these foreign widgets, the extension permits changes to the site. WebMakeup allows you to update embedded nodes with real data or not. Furthermore, inserted nodes can be collapsed by clicking on a user-defined node. MDWA [77] introduces a novel client-side and server-side component-based approach for developing WA applications. They suggest a model-driven approach to client-side and server-side creation that can increase the abstraction level. They provide a set of tools to design the composition of core applications. These apps provide new back-end features to help with WA. The main purpose of the front-end is to enhance the web page.

These last tools perform differently from MAWA since they act on the behavior of the web page. MAWA does not change or add new behavior to the web page. Focusing on content modifications, MDWA and CrowdMock allow the user to add elements already predefined by the system, as Omelette does. MAWA does not have any predefined elements. MDWA

also allows adding content from other pages, but it never updates its content. MDWA makes simple copies of HTML and CSS. MoWA adds information based on sensors such as GPS and adds information to a given web page depending on the location. This application focuses on pages such as museums or zoos. WebMakeup allows inserting content from other pages. This content can be dynamic using iframes, static through HTML and CSS copies (no updates of content allowed), and static-dynamic, whose content is changed based on certain content on the page (like looking up the rating of a certain movie in different sources). MAWA adds static content that can be updated if the user wants, but does not allow dynamic content to be added.

5.5 Behavior adaptation

Finally, some WA tools only include behavior modifications or additions. For example, to define the required native input, OFIE [78] uses Programming by Demonstration, which must be implemented automatically when the rule is triggered. This approach enables end-users to identify operations by simply executing the necessary native input interactions on the application's graphical user interface. The end-user is not required to have any programming experience or to write or modify any code. Cowpath [25] is a Firefox browser extension that, through a Domain-Specific Language, permits the creation of a "path" that connects related websites. The developer connects related web pages on a topic with each other, including a button that, when clicked, loads a new website related to the previous one. Finally, developments can be shared on social networks with other users. The change or inclusion of behavior by MAWA is not contemplated, which makes OFIE and Cowpath very different tools compared to MAWA.

In summary, Table 4 displays the characteristics of the presented tools. These tools have primarily been developed for two different platforms: mobile devices (8 tools) and desktops (9 tools). Most of the tools are client-side applications that are installed in the browser as a plugin or extension, or simply as tools installed on the client. MDWA and W3touch implement a hybrid approach, in which they require both a client and a server-side application to be executed. CrowdAdapt is the only tool deployed on a server. Regarding collaborative development, CrowdMock and MDWA have implemented a method that allows for augmentation development in a collaborative way with other users. Additionally, they permit sharing these creations with other users. Despite the fact that other tools do not support collaborative development, they have implemented methods to share their creations with other users. WA tools draw upon a variety of programming paradigms, including visual languages, spreadsheets, programming by demonstration, domain-specific languages (DSL), and model-based automation [79]. Only Sticklet and Cowpath have created a DSL that enables WA creations. Programming by demonstration is the second most commonly used paradigm, although visual programming is the most popular.

Regardless of the target platform of these applications, the execution mode, the subject of adaptation, or the programming paradigm, their ultimate objective is to allow users to adapt their favorite websites to their needs. In our approach, MAWA focuses on mobile browsing and is implemented as a Firefox extension (client-side). It is a visual programming tool whose primary objective is to enhance the browsing experience of end-users by adapting the layout of websites to their needs and by adding content from third-party websites via WA. All WA applications that modify the layout are designed for mobile, reflecting the importance of meeting users' needs and adapting to their requirements.

Table 4 Web augmentation tools classification

Name	Platform	Execution mode	Subject of adaptation	Collaborative	Programming paradigm	Reference
CrowdAdapt	Mobile	Server	Layout	Share with other	Programming by demonstration	[65]
UIWear	Mobile	Client	Layout	NO	Visual programming	[66]
W3touch	Mobile	Client-Server	Layout	NO	Visual programming	[67]
PageTailor	Mobile	Client	Layout	Share with other	Programming by demonstration	[68]
Collapse-to-Zoom	Mobile	Client	Layout	NO	Programming by demonstration	[69]
SAM	Mobile	Client	Layout Content	NO	Programming by demonstration	[70]
ALOE	Desktop	Client	Content	NO	Visual programming	[71]
Sticklet	Desktop	Client	Content	Share with other	DSL	[72]
ENIA	Desktop	Client	Content	Share with other	Visual programming	[73]
Excore	Desktop	Client	Content	NO	Visual programming	[24]
Omelette	Desktop	Client	Content	Share with other	Visual programming	[74]
CrowdMock	Desktop	Client	Content Behavior	Collaborative development and sharing	Visual programming	[75]
MoWA	Mobile	Client	Content Behavior	Share with other	Visual programming	[6]
WebMakeup	Desktop	Client	Content Behavior	Share with other	Visual programming	[76]
MDWA	Desktop	Client-Server	Content Behavior	Collaborative development and sharing	Visual programming	[77]
OFIE	Mobile	Client	Behavior	Share with other	Programming by demonstration	[78]
Cowpath	Desktop	Client	Behavior	Share with other	DSL	[25]
MAWA	Mobile	Client	Layout Content	NO	Visual programming	–

A DSL is a type of programming language characterized by a higher level of abstraction, tailor-made, and optimized for solving specific classes of problems [80]. On the other hand, Programming by Demonstration encompasses a range of approaches aimed at enabling end-users to program computers effectively, striving to access the full capabilities of computing technology [81]. When comparing these definitions to Visual Programming, which allows users to create programs by manipulating graphical program elements rather than specifying them in text [82], it becomes evident that DSLs and Programming by Demonstration tools differ significantly from Visual Programming tools. For this reason, when comparing MAWA with tools based on DSLs and Programming by Demonstration tools, it's clear that the techniques used are very different. Therefore, MAWA should be compared with Visual programming tools.

When considering Visual programming applications, we can find tools designed for desktop use, such as ALOE [71], ENIA [73], Excore [24], Omelette [74], CrowdMock [75], WebMakeup [76], and MDWA [77] and others designed for mobile platforms like UIWear [66], W3touch [67], and MoWA [6]. Among these 10 Visual programming tools, 7 are desktop-centric. The use of different devices leads to varying user interactions. However, each tool has a distinct design and enables the creation of adaptations in unique ways. In the case of ALOE, this tool allows the user to select text for translation into English. However, it does not offer the user the ability to perform other actions or select fonts or text from third-party pages, as MAWA does. Tools like ENIA, Omelette, CrowdMock, MDWA (for desktop), and MoWA (for mobile) have predefined components that can be used for web augmentation. Consequently, these tools do not facilitate the creation of elements with information from third-party sites, as MAWA does. In all of these tools, the end-user must employ the components created for augmentation. Excore, on the other hand, functions as a capture & replay tool and involves a special configuration step to select components from third-party sites. The process for selecting these components differs significantly from MAWA, as Excore replicates the user's actions and continuously provides updated information. None of these tools allows for the deletion or relocation of elements on the augmented page. In the case of WebMakeup, it is possible to extract information from third-party sources, delete, and move elements. The interaction method varies between desktop and mobile devices. However, WebMakeup inserts iframes of the components, enabling interaction with the extracted component while keeping the information updated without requiring background updates, as is done in MAWA. Among the mobile WA tools, UIWear only allows for the reorganization of page layouts, while W3touch detects and addresses user interaction problems by adjusting the layout to avoid these issues. These tools offer fewer functionalities compared to MAWA.

6 Conclusions and future work

The growth in the number of mobile browser users has obliged web developers to design websites and make them adaptable to various screen resolutions. The problem is that this adaptation is not free, and developers are still primarily designing for desktop users. This is the reason why mobile users often feel disadvantaged during their browsing experience. The main causes are the impossibility of having a different version of the same website for each user, and end-users not being able to modify the coding of their favorite websites. Additionally, the limited screen resolution complicates web interactions, leading to more scrolling, tapping, branching, and tab-switching by users. Consequently, the time required to complete actions is extended, resulting in increased battery consumption.

In order to solve these disadvantages, we have introduced MAWA, a Firefox mobile browser extension that allows users to reorder websites and integrate content from various sources. The primary features include the removal of unnecessary content, repositioning of content to minimize scrolling, and the option to add new content. This added content can be static if the required information remains constant, but it can also be automatically updated if users choose this option during the editing process. The latter option will replace the outdated external content with the most current information in the corresponding element.

An evaluation has been carried out to ensure that MAWA mitigates or resolves the mobile browsing drawbacks. First, a quantitative evaluation has been performed. In this evaluation we have demonstrated that MAWA reduces battery consumption by 87.6% and that users interactions (scrolling, tapping, branching etc.) are reduced by 95.39% reducing the time needed to complete actions by 92.75%. Nonetheless, MAWA increments the computational cost by 9.18%, which is affordable considering the previous improvements. As for the qualitative evaluation, a usability questionnaire was first accomplished. The subjects affirmed that MAWA is very easy to use and easy to learn. For the evaluation, the PSSUQ questionnaire has been used which shows that the system quality, information quality and interface quality of MAWA are very satisfactory. Finally, the causes and consequences described in the Sect. 2 have been assessed. Improving the screen size is impossible for a browser extension, nevertheless, the subjects claim that MAWA helps to solve all these issues. MAWA increases the computational cost, but the increase (9.18%) is not considered a major drawback, according to the interviewed participants.

In the future, we plan to develop a debugging assistant that will facilitate end-user debugging of the resulting applications. The aim of this assistant will be to help end-users detect failures and the reasons behind them. Moreover, the MAWA assistant will assist end-users in repairing malfunctions.

Another enhancement would involve sharing augmentations with other MAWA users. This feature might be essential for increasing the extension's usage. Adding a new button to MAWA's menu would be an option to upload basic augmentation information to a repository. This augmentation could then be downloaded to reproduce it in another browser, enabling the creation of a MAWA community. Debugging features would help customize another member's augmentations to adapt them to the user's needs.

We would also like this extension to be available on Android. Currently, Android does not allow extensions to be installed, but there may be a predisposition to add this option in the future. If, as with desktop computer browsers, the extensions are compatible with both browsers, the adaptation work will be minimal. Android is used by 65%⁶ of users, which could significantly increase MAWA's usage.

Website upgrades are the major challenge for WA tools, as nodes may no longer be accessible. Based on [83–85], the idea is to create a mechanism capable of regenerating Xpaths for WA tools. When one of the Xpaths fails to find an element, the algorithm would use information from the previous node to repair the broken Xpaths.

Acknowledgements This work was carried out by the Software and Systems Engineering research group of Mondragon Unibertsitatea, supported by the Department of Education, Universities and Research of the Basque Government (IT1326-19).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

⁶ <https://gs.statcounter.com/browser-market-share/mobile/worldwide>.

References

1. Nurshuhada A, Yusop ROM, Azmi A, Ismail SA, Sarkan HM, Kama N (2019) Enhancing performance aspect in usability guidelines for mobile web application. In: 2019 6th international conference on research and innovation in information systems (ICRIIS). IEEE, pp 1–6
2. García-Lopez E, García-Cabot A, Manresa-Yee C, de Marcos L, Pagés-Arévalo C (2017) Validation of navigation guidelines for improving usability in the mobile web. *Comput Stand Interfaces* 52:51–62
3. Johnson TA, Seeling P (2014) Desktop and mobile web page comparison: characteristics, trends, and implications. *IEEE Commun Mag* 52(9):144–151
4. Boehm BW, Clark B, Horowitz E, Westland JC, Madachy RJ, Selby RW (1995) Cost models for future software life cycle processes: COCOMO 2.0. *Ann Softw Eng* 1:57–94
5. Scaffidi C, Shaw M, Myers B (2005) Estimating the numbers of end users and end user programmers. In: 2005 IEEE symposium on visual languages and human-centric computing (VL/HCC'05). IEEE, pp 207–214
6. Bosetti G, Firmenich S, Gordillo SE, Rossi G, Winckler M (2017) An end user development approach for mobile web augmentation. *Mob Inf Syst* 2017:2525367–1252536728
7. Bouvin NO (1999) Unifying strategies for web augmentation. In: Proceedings of the tenth ACM conference on hypertext and hypermedia: returning to our diverse roots, pp 91–100
8. Aldalur I, Winckler M, Díaz O, Palanque P (2017) Web augmentation as a promising technology for end user development. In: New perspectives in end-user development. Springer, pp 433–459
9. Myers BA (1986) Visual programming, programming by example, and program visualization: a taxonomy. *ACM Sigchi Bull* 17(4):59–66
10. Lieberman H, Paternò F, Wulf V (eds) (2006) End user development. Human–computer interaction series. Springer, Berlin
11. Spahn M, Dörner C, Wulf V (2008) End user development: approaches towards a flexible software design. In: 16th european conference on information systems, ECIS 2008, Galway, Ireland, 2008, pp 303–314
12. Frain B (2012) Responsive web design with HTML5 and CSS3. Packt Publishing Ltd, Birmingham
13. Pinandito A, Az-zahra HM, Fanani L, Putri AV (2017) Analysis of web content delivery effectiveness and efficiency in responsive web design using material design guidelines and user centered design. In: International conference on sustainable information engineering and technology (SIET). IEEE, pp 435–441
14. Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S (2008) A design science research methodology for information systems research. *J Manag Inf Syst* 24(3):45–77
15. Venable J, Baskerville R (2012) Eating our own cooking: toward a more rigorous design science of research methods. *Electron J Bus Res Methods* 10(2):141–153
16. Díaz O, Contell JP, Venable JR (2017) Strategic reading in design science: let root-cause analysis guide your readings. In: Designing the digital transformation: 12th international conference, DESRIST 2017, Karlsruhe, Germany, May 30–June 1, 2017, Proceedings. Lecture notes in computer science, vol 10243. Springer, pp 231–246
17. Wroblewski L (2012) Mobile first, vol 6. Editions Eyrolles, Paris
18. Punchoojit L, Hongwarittorn N (2017) Usability studies on mobile user interface design patterns: a systematic literature review. *Adv Hum Comput Interact* 2017:6787504–1678750422
19. Marcotte E (2017) Responsive web design: a book apart N 4. Editions Eyrolles, Paris
20. Mahajan S, Abolhassani N, McMin P, Halfond WGJ (2018) Automated repair of mobile friendly problems in web pages. In: Proceedings of the 40th international conference on software engineering, ICSE 2018, Gothenburg, Sweden, May 27–June 03, 2018. ACM, pp 140–150
21. Nebeling M, Norrie MC (2013) Responsive design and development: methods, technologies and current issues. In: Web engineering: 13th international conference, ICWE 2013, Aalborg, Denmark, 8–12 July 2013. Proceedings, vol 7977. Springer, pp 510–513
22. Lavie T, Sela M, Oppenheim I, Inbar O, Meyer J (2010) User attitudes towards news content personalization. *Int J Hum Comput Stud* 68(8):483–495
23. Rhodes BJ (2000) Margin notes: building a contextually aware associative memory. In: Proceedings of the 5th international conference on intelligent user interfaces, IUI 2000, New Orleans, LA, USA, 9–12 January 2000. ACM, pp 219–224
24. Aldalur I, Perez A, Larrinaga F (2019) Customizing websites through automatic web search. In: Human–computer interaction: INTERACT 2019: 17th IFIP TC 13 international conference, Paphos, Cyprus, 2–6 Sept 2019, Proceedings, Part II, vol 11747. Springer, pp 598–618
25. Díaz O, Sosa JD, Trujillo S (2013) 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, 02–04 May 2013, pp 69–78

26. Castaneda L, Villegas NM, Müller HA (2014) Self-adaptive applications: on the development of personalized web-tasking systems. In: 9th international symposium on software engineering for adaptive and self-managing systems, SEAMS 2014, Proceedings, Hyderabad, India, 2–3 June 2014. ACM, pp 49–54
27. Winckler M, Cava RA, Barboni E, Palanque PA, Freitas CMDS (2015) Usability aspects of the inside-in approach for ancillary search tasks on the web. In: Human–computer interaction—INTERACT 2015—15th IFIP TC 13 international conference, Bamberg, Germany, 14–18 Sept 2015, Proceedings, Part II. Lecture notes in computer science, vol 9297. Springer, pp 211–230
28. Lugtig P, Toepoel V (2016) The use of pcs, smartphones, and tablets in a probability-based panel survey: effects on survey measurement error. *Soc Sci Comput Rev* 34(1):78–94
29. Syed ZA, Helmick J, Banerjee S, Cukic B (2019) Touch gesture-based authentication on mobile devices: the effects of user posture, device size, configuration, and inter-session variability. *J Syst Softw* 149:158–173
30. Peytchev A, Hill CA (2010) Experiments in mobile web survey design: similarities to other modes and unique considerations. *Soc Sci Comput Rev* 28(3):319–335
31. Wobbrock JO, Forlizzi J, Hudson SE, Myers BA (2002) Webthumb: interaction techniques for small-screen browsers. In: Proceedings of the 15th annual ACM symposium on user interface software and technology, Paris, France, 27–30 Oct 2002. ACM, pp 205–208
32. Radecký M, Smutný P (2014) Evaluating user reaction to user interface element using eye-tracking technology. In: Proceedings of the 2014 15th international Carpathian control conference (ICCC). IEEE, pp 475–480
33. Aula A, Khan RM, Guan Z (2010) 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, 10–15 Apr 2010. ACM, pp 35–44
34. Lagun D, Hsieh C, Webster D, Navalpakkam V (2014) Towards better measurement of attention and satisfaction in mobile search. In: The 37th international ACM SIGIR conference on research and development in information retrieval, SIGIR '14, Gold Coast, QLD, Australia, 06–11 July 2014. ACM, pp 113–122
35. Liu C, Liu C, Mao H, Su W (2019) Tilt-scrolling: a comparative study of scrolling techniques for mobile devices. In: Intelligent computing methodologies—15th international conference, ICIC 2019, Nanchang, China, 3–6 Aug 2019, Proceedings, Part III, vol 11645. Springer, pp 189–200
36. Ronkainen S, Häkkinen J, Kaleva S, Colley A, Linjama J (2007) Tap input as an embedded interaction method for mobile devices. In: Proceedings of the 1st international conference on tangible and embedded interaction 2007, Baton Rouge, Louisiana, USA, 15–17 Feb 2007. ACM, pp 263–270
37. Athukorala K, Glowacka D, Jacucci G, Oulasvirta A, Vreeken J (2016) Is exploratory search different? A comparison of information search behavior for exploratory and lookup tasks. *J Assoc Inf Sci Technol* 67(11):2635–2651
38. Lehmann F, Kipp M (2018) How to hold your phone when tapping: a comparative study of performance, precision, and errors. In: Proceedings of the 2018 ACM international conference on interactive surfaces and spaces, ISS 2018, Tokyo, Japan, 25–28 Nov 2018. ACM, pp 115–127
39. Huang J, White RW (2010) Parallel browsing behavior on the web. In: HT'10, Proceedings of the 21st ACM conference on hypertext and hypermedia, Toronto, Ontario, Canada, 13–16 June 2010, pp 13–18
40. Zhang H, Zhao S (2011) Measuring web page revisitation in tabbed browsing. In: Proceedings of the international conference on human factors in computing systems, CHI 2011, Vancouver, BC, Canada, 7–12 May 2011, pp 1831–1834
41. Chang JC, Hahn N, Kim Y, Coupland J, Breneisen B, Kim HS, Hwang J, Kittur A (2021) When the tab comes due: challenges in the cost structure of browser tab usage. In: Proceedings of the 2021 CHI conference on human factors in computing systems, pp 1–15
42. Raghavan S, Parampalli U, Raghavan SV (2017) Re-engineering simultaneous internet sessions process-separated browsers. In: Proceedings of the Australasian computer science week multiconference, ACSW 2017, Geelong, Australia, January 31–February 3, 2017, pp 70–17010
43. Huang J, Lin T, White RW (2012) No search result left behind: branching behavior with browser tabs. In: Proceedings of the fifth international conference on web search and web data mining, WSDM 2012, Seattle, WA, USA, 8–12 Feb 2012. ACM, pp 203–212
44. Hahn N, Chang JC, Kittur A (2018) Bento browser: complex mobile search without tabs. In: Proceedings of the 2018 CHI conference on human factors in computing systems, CHI 2018, Montreal, QC, Canada, 21–26 Apr 2018. ACM, p 251
45. Tahir R, Raza A, Ahmad F, Kazi J, Zaffar F, Kanich C, Caesar M (2018) 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, 16–19 Apr 2018. IEEE, pp 2618–2626
46. Al-Khalifa AS, Al-Khalifa HS (2012) Emerging URL patterns in mobile websites: a preliminary results. In: Proceedings of the 3rd international conference on ambient systems, networks and technologies (ANT

- 2012), the 9th international conference on mobile web information systems (MobiWIS-2012), Niagara Falls, Ontario, Canada, 27–29 Aug 2012. *Procedia computer science*, vol 10. Elsevier, pp 952–959
47. Cao X, Masood A, Luqman A, Ali A (2018) Excessive use of mobile social networking sites and poor academic performance: antecedents and consequences from stressor-strain-outcome perspective. *Comput Hum Behav* 85:163–174
48. D'Ambrosio S, De Pasquale S, Iannone G, Malandrino D, Negro A, Patimo G, Scarano V, Spinelli R (2016) Energy consumption and privacy in mobile web browsing: individual issues and connected solutions. *Sustain Comput Inform Syst* 11:63–79
49. Javed A, Shahid M, Sharif M, Yasmin M (2017) Energy consumption in mobile phones. *Int J Comput Netw Inf Secur* 9:18–28
50. Lewerentz A, Lindvall J (2012) Performance and energy optimization for the android platform. *Blekinge*
51. Cody-Kenny B, Manganiello U, Farrelly J, Ronayne A, Considine E, McGuire T, O'Neill M (2018) Investigating the evolvability of web page load time. In: *International conference on the applications of evolutionary computation*. Springer, pp 769–777
52. Pourghassemi B, Sani AA, Chandramowlishwaran A (2019) What-if analysis of page load time in web browsers using causal profiling. *Proc ACM Meas Anal Comput Syst* 3(2):27–12723
53. Mustonen T, Olkkonen M, Häkkinen J (2004) Examining mobile phone text legibility while walking. In: *Conference on human factors in computing systems, CHI 2004, Vienna, Austria, 24–29 Apr 2004*. ACM, pp 1243–1246
54. Andreadis I (2015) Comparison of Response Times between Desktop and Smartphone Users. In: Toninelli, D et al (eds.), *Mobile Research Methods*. London: Ubiquity Press. <https://doi.org/10.5334/bar.e>
55. Yesilada Y, Harper S, Chen T, Trewin S (2010) Small-device users situationally impaired by input. *Comput Hum Behav* 26(3):427–435
56. Setlur V, Rossoff S, Gooch B (2011) Wish I hadn't clicked that: context based icons for mobile web navigation and directed search tasks. In: *Proceedings of the 16th international conference on intelligent user interfaces, IUI 2011, Palo Alto, CA, USA, 13–16 Feb 2011*. ACM, pp 165–174
57. Xie X, Wang C, Chen L-Q, Ma W-Y (2005) An adaptive web page layout structure for small devices. *Multimed Syst* 11(1):34–44
58. Walsh TA, Kapfhammer GM, McMinn P (2017) Automated layout failure detection for responsive web pages without an explicit oracle. In: *Proceedings of the 26th ACM SIGSOFT international symposium on software testing and analysis, Santa Barbara, CA, USA, 10–14 July 2017*. ACM, pp 192–202
59. Aldalur I, Perez A, Larrinaga F (2021) 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. Lecture notes in computer science*, vol 12935. Springer, pp 221–242
60. Maguire M, Tang M (2014) Comparison test of website use with mobile phone and laptop computer. In: *Human-computer interaction. Applications and services—16th international conference, HCI international 2014, Heraklion, Crete, Greece, 22–27 June 2014, Proceedings, Part III. Lecture notes in computer science*, vol 8512. Springer, pp 146–154
61. Ricca F, Leotta M, Stocco A, Clerissi D, Tonella P (2013) Web testware evolution. In: *15th IEEE international symposium on web systems evolution, WSE 2013, Eindhoven, The Netherlands, 27 Sept 2013*, pp 39–44
62. Lewis JR (1992) Psychometric evaluation of the post-study system usability questionnaire: the pssuq. In: *Proceedings of the human factors society annual meeting*, vol 36. Sage Publications, Los Angeles, pp 1259–1260
63. Lewis JR (1995) IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *Int J Hum Comput Interact* 7(1):57–78
64. Peltonen E, Lagerspetz E, Nurmi P, Tarkoma S (2016) Constella: crowdsourced system setting recommendations for mobile devices. *Pervasive Mob Comput* 26:71–90
65. Nebeling M, Speicher M, Norrie MC (2013) Crowdadapt: enabling crowdsourced web page adaptation for individual viewing conditions and preferences. In: *ACM SIGCHI symposium on engineering interactive computing systems, EICS'13, London, United Kingdom, 24–27 June 2013*. ACM, pp 23–32
66. Xu J, Cao Q, Prakash A, Balasubramanian A, Porter DE (2017) Uiwear: easily adapting user interfaces for wearable devices. In: *Proceedings of the 23rd annual international conference on mobile computing and networking, MobiCom 2017, Snowbird, UT, USA, 16–20 Oct 2017*. ACM, pp 369–382
67. Nebeling M, Speicher M, Norrie MC (2013) W3touch: metrics-based web page adaptation for touch. In: *2013 ACM SIGCHI conference on human factors in computing systems, CHI '13, Paris, France, Apr 27–May 2, 2013*. ACM, pp 2311–2320

68. Bila N, Ronda T, Mohamed I, Truong KN, de Lara E (2007) Pagetailor: reusable end-user customization for the mobile web. In: Proceedings of the 5th international conference on mobile systems, applications, and services (MobiSys 2007), San Juan, Puerto Rico, 11–13 June 2007. ACM, pp 16–29
69. Baudisch P, Xie X, Wang C, Ma W (2004) Collapse-to-zoom: viewing web pages on small screen devices by interactively removing irrelevant content. In: Feiner S, Landay JA (eds) Proceedings of the 17th annual ACM symposium on user interface software and technology, Santa Fe, NM, USA, 24–27 Oct 2004. ACM, pp 91–94
70. Gobert C, Todi K, Bailly G, Oulasvirta A (2019) SAM: a modular framework for self-adapting web menus. In: Proceedings of the 24th international conference on intelligent user interfaces, IUI 2019, Marina del Ray, CA, USA, 17–20 Mar 2019. ACM, pp 481–484
71. Trusty A, Truong KN (2011) Augmenting the web for second language vocabulary learning. In: Proceedings of the international conference on human factors in computing systems, CHI 2011, Vancouver, BC, Canada, 7–12 May 2011. ACM, pp 3179–3188
72. Díaz O, Arellano C, Azanza M (2013) A language for end-user web augmentation: caring for producers and consumers alike. ACM Trans Web 7(2):9–1951
73. Fernández-García AJ, Iribarne L, Corral A, Criado J, Wang JZ (2018) A flexible data acquisition system for storing the interactions on mashup user interfaces. Comput Stand Interfaces 59:10–34
74. Chudnovskyy O, Nestler T, Gaedke M, Daniel F, Fernández-Villamor JI, Chepegin VI, Fornas JA, Wilson S, Kögler C, Chang H (2012) End-user-oriented telco mashups: the OMELETTE approach. In: Proceedings of the 21st world wide web conference, WWW 2012, Lyon, France, 16–20 Apr 2012 (Companion Volume), pp 235–238
75. Firmenich D, Firmenich S, Rivero JM, Antonelli L, Rossi G (2018) Crowdmock: an approach for defining and evolving web augmentation requirements. Requir Eng 23(1):33–61
76. Díaz O, Arellano C, Aldalur I, Medina H, Firmenich S (2014) End-user browser-side modification of web pages. In: Web information systems engineering—WISE 2014—15th international conference, Thessaloniki, Greece, 12–14 Oct 2014, Proceedings, Part I. Lecture notes in computer science, vol 8786. Springer, pp 293–307
77. Urbietta M, Firmenich S, Bosetti G, Maglione P, Rossi G, Olivero MA (2020) MDWA: a model-driven web augmentation approach-coping with client- and server-side support. Softw Syst Model 19(6):1541–1566
78. Bellal Z, Elouali N, Benslimane SM, Acarturk C (2020) Integrating mobile multimodal interactions based on programming by demonstration. Int J Hum Comput Interact 37(5):418–433
79. Aghaee S, Pautasso C (2011) End-user programming for web mashups: open research challenges. In: Current trends in web engineering—workshops, doctoral symposium, and tutorials, Held at ICWE 2011, Paphos, Cyprus, 20–21 June 2011. Revised selected papers. Lecture notes in computer science, vol 7059. Springer, pp 347–351
80. Hudak P (1998) Modular domain specific languages and tools. In: Proceedings. Fifth international conference on software reuse (Cat. No. 98TB100203). IEEE, pp 134–142
81. Cypher A, Halbert DC (1993) Watch what I do: programming by demonstration. MIT press, Cambridge
82. Burnett MM, McIntyre DW (1995) Visual programming. Computer-Los Alamitos 28:14
83. Aldalur I, Larrinaga F, Perez A (2020) ABLA: an algorithm for repairing structure-based locators through attribute annotations. In: Web information systems engineering-WISE 2020-21st international conference, Amsterdam, The Netherlands, 20–24 Oct 2020, Proceedings, Part II. Lecture notes in computer science, vol 12343. Springer, pp 101–113
84. Kirinuki H, Tanno H, Natsukawa K (2019) 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, 24–27 Feb 2019, pp 310–320
85. Hammoudi M, Rothermel G, Stocco A (2016) 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, 13–18 Nov 2016, pp 751–762

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



and impactful contributions to the field.

Dr. Iñigo Aldalur received the B.Sc. degree in computer science from the University of the Basque Country, Spain, in 2009, and the Ph.D. degree from the Department of Computer Languages and Systems, University of the Basque Country, in 2017. He is currently a member of the Information Systems Research Group, Department of Electronics and Computer Science. Dr. Aldalur's academic journey has been marked by a deep commitment to advancing knowledge and innovation in the field of computer science. His research interests span a wide range of topics, with a particular focus on software engineering, web engineering, and human-computer interaction. Throughout his career, he has contributed significantly to the development of methodologies and tools aimed at improving software quality, enhancing web application performance, and facilitating seamless interactions between users and digital systems. Dr. Aldalur's dedication to interdisciplinary collaboration and his passion for exploring the intersection of technology and human experience continue to drive his pursuit of cutting-edge research



and Mobile Developments (both web and native).

Dr. Alain Perez is a Lecturer & Researcher in the IT department of MGEP-MU Computer Science Engineer (2010) and PhD in Computer Science with the thesis "Semantic Web and Semantic Technologies to Enhance Innovation and Technology Watch Process", also by MGEP-MU (2016). Since 2009 he has worked on research projects in Innovation, Technology Watch and Competitive Intelligence contributing to Web technologies, Semantic Web and Machine Learning techniques for automatic content classification. He has lectured classes in subjects such as Advanced Software Architectures, Web Engineering, Computer Basics or Improvement and Maintenance of Embedded Systems. Currently, he also participates in Smart City projects (such as CityFied, SmarEnCity, Rennovates...) contributing in the field of Interoperability (AMQP, ModBus, EEBus, Linked Data, Asset AdministrationShell...), Web Page Development (Flask, Symfony, Spring...) and Database generation and maintenance (MySQL, SQLServer, SQLite...). His main areas of interest are: interoperability, Web Engineering, Semantic Web



Engineering in 2010. Expertise: Web engineering, interoperability, web architecture, semantic web

Dr. Felix Larrinaga is currently a lecturer and a researcher at Mondragon University (Basque Country, Spain). Since 2000, he combines lecturing and research at Mondragon University with an extensive experience in industrial work as a project leader. He has actively participated in the creation of the new research team on Web Engineering at the Faculty of Engineering. The research activities of the group are in the areas of interoperability, web architecture, microservices, and semantic web technologies. The team works on multidisciplinary projects with other universities, technological centers and enterprises in areas such energy, health or industry. At the moment, Dr Larrinaga is working on several projects related to manufacturing, smart cities and energy (*QU4LITY*, *Pliades*, *DiManD*, *ArrowheadTools*, *DropandSmartEnCity*). Dr Larrinaga obtained his PhD in Mobile Communications at Staffordshire University (UK) in 1999. Previously, he finished the BSc (Hons) in Industrial Electronics at Mondragon University in 1995. He finished the MSc in Computing



Dr. Miren Illarramendi studied Telecommunications Engineering at the University of the Basque Country, completing her studies in 2002. During the years 2001-2002, she conducted research activities at the Ikerlan S.Coop research center in the Communications department. Since September 2002, she has been working in the Department of Informatics and Electronics at Mondragon Unibertsitatea. From 2007 to 2016, she served as the coordinator of the Distributed and Real-Time Systems Research Group, and since 2021, she has been coordinating the Software Engineering and Systems research group. She has actively participated in several European projects (SafeCer, Arrowhead, Productive 4.0, Adeptness, etc.). Currently, she also coordinates the DiManD Innovative Training Network (ITN) research project under the Marie Skłodowska-Curie Innovative Training Networks call. Illarramendi's research focuses on Critical Systems and Reliability, Distributed Systems, and remote monitoring and control of CPS. In 2019, she obtained her Ph.D. in Computer Science with the title "Runtime Observable and

Adaptable UML State Machine-Based Software Components Generation" from Mondragon University, with the qualification of Summa Cum Laude. In terms of academic training activities, Miren has been a lecturer for courses such as Fundamentals of Programming, Advanced Programming, Operating Systems, and Distributed Systems in the Computer Science degree. Additionally, she has taught the Reliability and Performance Analysis course in the Master's program in Embedded Systems and currently teaches the Platforms and Infrastructures course in the Master's program in Data Analysis and Cloud Computing (for the last 4 years). Regarding collaboration with companies, in recent years Miren has participated in multiple projects in collaboration with companies, and she is currently involved in a project with Sener Aerospace where she spends 2 days a week at the company to carry out transfer activities related to a project on Model Based Systems Engineering, implementing this methodology and adapting the necessary technology for it.