Computer Software and Media Applications 2024, 7(1), 6736.
https://doi.org/10.24294/csma.v7i1.6736

Article

Mockplug: A high-fidelity mocking tool for plugging functional requirements into existing web applications

Diego Firmenich[1,*], Leonardo Morales[1,2,3], Gastón Mura[1], Nicolás Calfuquir[1]

[1] DIT, Departamento de Informática Trelew, Facultad de Ingeniería, Universidad Nacional de la Patagonia, Comodoro Rivadavia U9100, Argentina
[2] Patagonian Institute of Social and Human Sciences (IPCSH), Centro Nacional Patagónico, National Council for Scientific and Technical Research, Puerto Madryn U9120, Argentina
[3] Imaging Science Laboratory, Department of Electrical and Computer Engineering, Universidad Nacional del Sur, Bahía Blanca B8000, Argentina
*** Corresponding author:** Diego Firmenich, dafirmenich@ing.unp.edu.ar

CITATION

Firmenich D, Morales L, Mura G, Calfuquir N. Mockplug: A high-fidelity mocking tool for plugging functional requirements into existing web applications. Computer Software and Media Applications. 2024; 7(1): 6736.
https://doi.org/10.24294/csma.v7i1.6736

ARTICLE INFO

Received: 29 May 2024
Accepted: 19 June 2024
Available online: 5 September 2024

Abstract: Mockplug is a browser extension that allows end users to specify their requirements for any existing web application through high-fidelity mockups. These mockups are built based on web augmentation techniques. This new way of specifying the requirements contains the intrinsic potential that the mockup is built on top of the application itself with elements of the same nature, containing technical information about the requirements in relation to application components from its origin automatically. This has great potential when it comes to being used as an input during the software development process. In this article, we disclose and describe the use and potential of this tool in two totally different approaches to building web software.

Keywords: web engineering; web applications; requirements engineering

1. Introduction

The use of mockups and prototyping techniques in software development has been a longstanding practice, dating back several decades. These methods have been recognized for their effectiveness in facilitating communication between engineers and end-users [1,2].

In recent years, there has been a shift towards more user-centric approaches, where end-users are actively involved in the requirements definition process. Tools that allow end-users to express their needs using text and layout tools have gained prominence [3,4]. These tools empower end-users to visually specify their requirements, leading to a better understanding of their needs by the development team.

Some model-driven approaches, such as Mockup-driven development [5], propose the evolution of prototypes into more formal models. However, despite their potential benefits, the adoption of such approaches is not widespread due to methodological challenges [6].

Overall, while mockups and prototyping remain valuable techniques in software development, the trend is towards greater involvement of end-users in the requirements definition process, facilitated by user-friendly tools that enable them to express their needs more effectively.

Mockplug is an innovative tool that empowers end-users to create high-fidelity mockups directly within existing web applications. By leveraging the existing application as a canvas, Mockplug streamlines the mockup creation process and facilitates more accurate representation of user needs.

One of the key advantages of Mockplug is its augmentation techniques, which enable end-users to interact with the web application in a familiar environment. By allowing users to select, remove, move, and add elements within the application, Mockplug ensures that the mockups closely align with the user's requirements and preferences. Moreover, the ability to collect components from different parts of the same application or even from different web applications enhances the flexibility and customization options available to end-users. This feature enables users to leverage existing design elements efficiently and facilitates the creation of comprehensive and cohesive mockups.

Overall, Mockplug is a valuable tool for enhancing the collaboration between end-users and developers in the software development process. By providing users with the means to visually specify their needs directly within the application interface, Mockplug facilitates clearer communication and promotes a more user-centric approach to design and development.

Compared to traditional mockup methods, this approach offers significant advantages. Firstly, users are spared the tedious task of starting their designs from scratch or constantly updating them with each app version. This saves time and effort, allowing for a more efficient workflow.

Secondly, these new artifacts for requirement definition not only make the process quicker and more visually intuitive, but also offer a unique advantage. By using the same elements as the application itself, they can potentially incorporate functionalities that directly interact with the original software. This means that users can create mockups that not only represent the look and feel of the final product but also simulate its functionality to some extent.

From the developer's perspective, this represents a paradigm shift in how requirements are defined and understood. It opens up new possibilities for collaboration and innovation. Developers can gain insights into user needs more directly, leading to more accurate and effective software development. Additionally, depending on the nature of the software being developed, this approach can offer various benefits. For instance, in complex projects, it can lead to better alignment between design and development teams, reducing the risk of misunderstandings or misinterpretations of requirements.

In the realm of traditional web application development, integrating this type of requirement definition into the process offers notable advantages. It simplifies requirements elicitation, making it easier for developers to understand and implement user needs. Additionally, it has the potential to influence developers' integrated development environments (IDEs), streamlining the development process further [7].

In the context of web augmentation artifacts created by end users, exemplified by platforms like Crowdmock [8], these requirements serve a crucial role. Users, even those without extensive programming skills, can construct and share software using various tools. One significant advantage is the automatic generation of source code, which includes all references to the relevant elements of interest in the DOM tree. We refer to these references as DOM Elements of Interest (DEOIs). This eliminates the need for users to manually search for these references in the Document Object Model (DOM), enhancing efficiency and usability.

This offers the advantage of automatically generating source code with all DEOIs, eliminating the need for users to manually search for these references on the target site and hard-code them in the source. Furthermore, this automation opens up the possibility for end users of the artifact to actively participate in the ongoing maintenance of the artifact, including updating DEOI references [9], which is often required when elements of the DOM Tree of web applications change for any reason.

Since Mockplug is a browser extension, users simply need to visit the extension marketplace and install it in their browser. Currently, it is available exclusively for the Chrome browser and is seamlessly integrated with the Trello application. This integration allows users to request permission to add stories to the Kanban board lists of developers who own the web applications of interest. Additionally, there are two discontinued branches for the Firefox browser in the repository. These branches, instead of being integrated with Trello, are integrated with the CrowdMock platform [8].

In this article, we present this tool in a stable experimental version. In section 2, we describe the fundamental background that gives rise to its operation. In section 3, we outline its architecture, main functionalities, and how to obtain the software for evaluation. In section 4, we provide two examples of usage in two completely different scenarios where we have conducted experiments. Finally, in section 5, we incorporate a discussion section on relevant aspects, followed by the conclusions in section 6.

2. Web augmentation background

The web augmentation technique is an approach to software development that dispenses with the creation of a new application, opting instead for the modification and enhancement of an existing website. Through this methodology, additional content is superimposed, the design is altered and the site's navigation capabilities are expanded, with the purpose of adapting and personalizing the user's experience, in order to improve performance and satisfaction [10].

We should think of this alteration as a layer over the original site, thus highlighting the non-intrusive character of the technique [11].

Web augmentation helps to meet user needs that were not originally identified or taken into account during web page design [12]. The adaptations made on web pages (in content, style or behavior) can take place on the web server, using a special proxy, or by modifying the web pages displayed in the web browser. The second case, called client-side, is the most common. Regarding the collaborative aspect, a web augmentation tool can be personal, can be shared and edited with other users, asynchronously or synchronously. The second approach (collaborative development) has gained greater prominence in recent years.

In the paper of Aldalur et al. [11], the authors highlight the benefits of using WA tools in End-User Development environments, referred to as "programming to achieve the result of a program primarily for personal, rather than public use" [13]. This is because web augmentation helps to realize this vision in the context of the web, as it helps to meet user needs that were not identified or considered during the initial design of the website. At the center of this dynamic, initiatives emerge such as the Web platform proposed for modeling web augmentations that abstract the back-end complexity for users of client-server applications [14]. As a result, the authors developed a tool that provides an end-to-end web experience for the design and execution of Web Augmentations, which require both client-side and server-side components.

3. Software description

3.1. Software architecture

As a browser extension, Mockplug comprises three main components that interact with each other through asynchronous messaging, as illustrated in **Figure 1**. The primary interface of the application is housed within the popup component, which constructs and presents the main user interface.

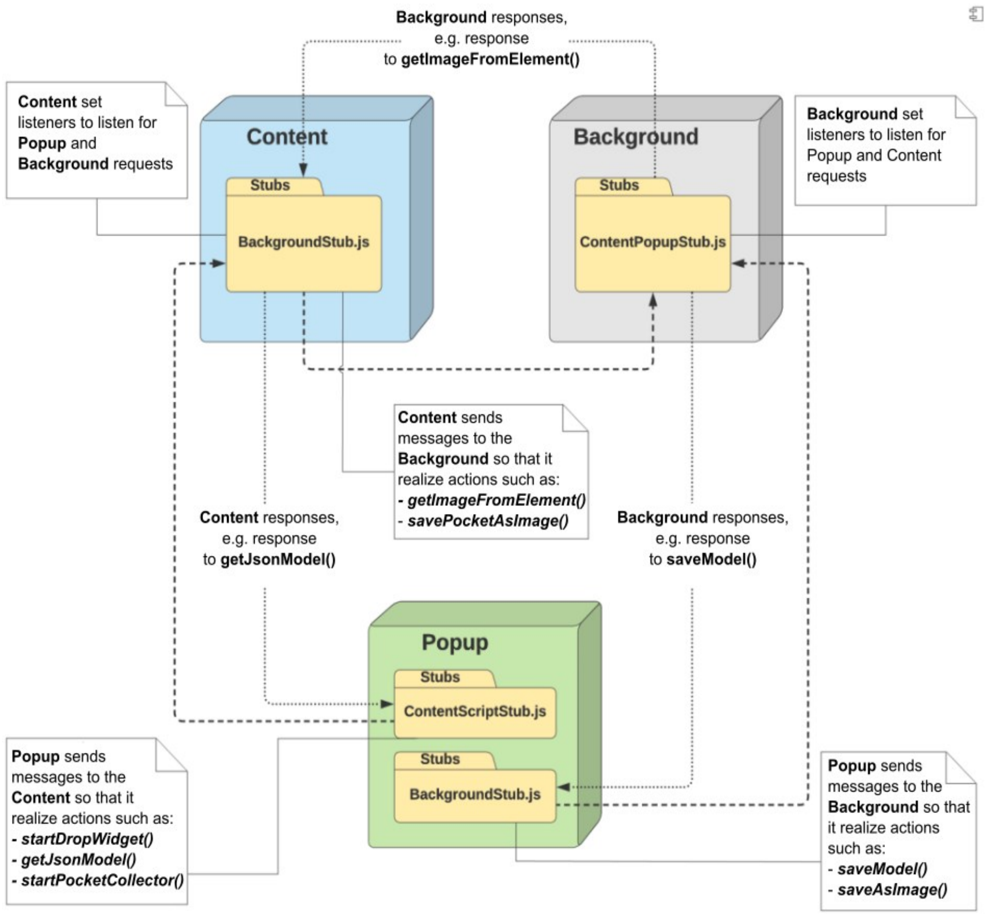

Background responses, e.g. response to getImageFromElement()
Content set listeners to listen for Popup and Background requests
Content
Stubs
BackgroundStub.js
Background
Stubs
ContentPopupStub.js
Background set listeners to listen for Popup and Content requests
Content sends messages to the Background so that it realize actions such as: - getImageFromElement() - savePocketAsImage()
Content responses, e.g. response to getJsonModel()
Background responses, e.g. response to saveModel()
Popup
Stubs
ContentScriptStub.js
Stubs
BackgroundStub.js
Popup sends messages to the Content so that it realize actions such as: - startDropWidget() - getJsonModel() - startPocketCollector()
Popup sends messages to the Background so that it realize actions such as: - saveModel() - saveAsImage()

Figure 1. Message handling examples between the main Mockplug components.

The other two fundamental components are the background and the content. The background component handles tasks such as image processing, preference management, and communication with the Trello REST API. Conversely, the content component is responsible for interacting with the augmented web application through its DOM tree. Both components initialize listeners to await incoming messages from other components.

There is a message exchange between the content and background components, but only the content component requests the background to execute certain actions and return results. Conversely, there are no requests from the background component to the content component. Unlike the background and content components, the popup component does not listen for messages. Instead, it serves as the graphical interface of the extension and triggers actions for the other components.

Figure 2 illustrates the Mockplug class diagram, showcasing the implementation of design patterns such as the factory method for creating different widget types, strategy for handling diverse methods of element insertion into the DOM tree of the specified web application, and command for seamless undo and redo functionalities.

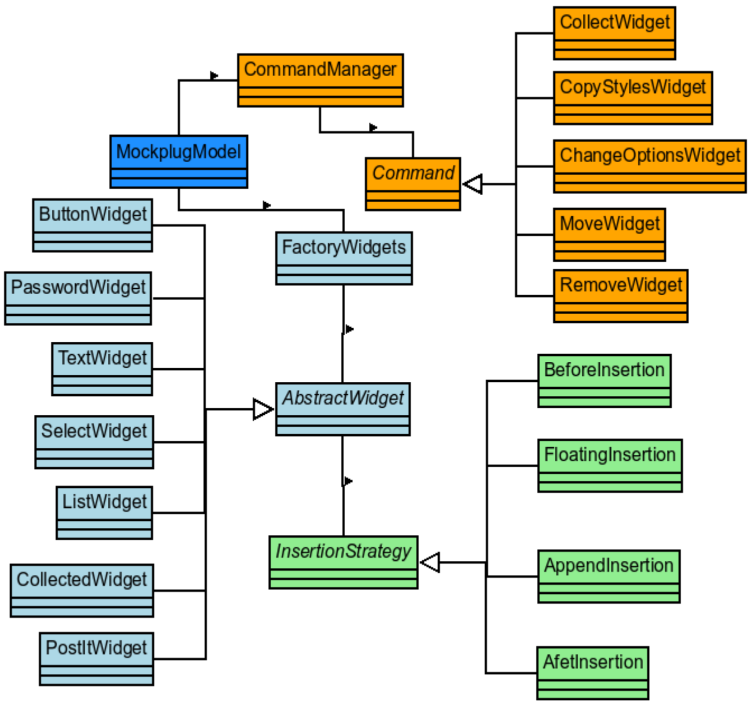
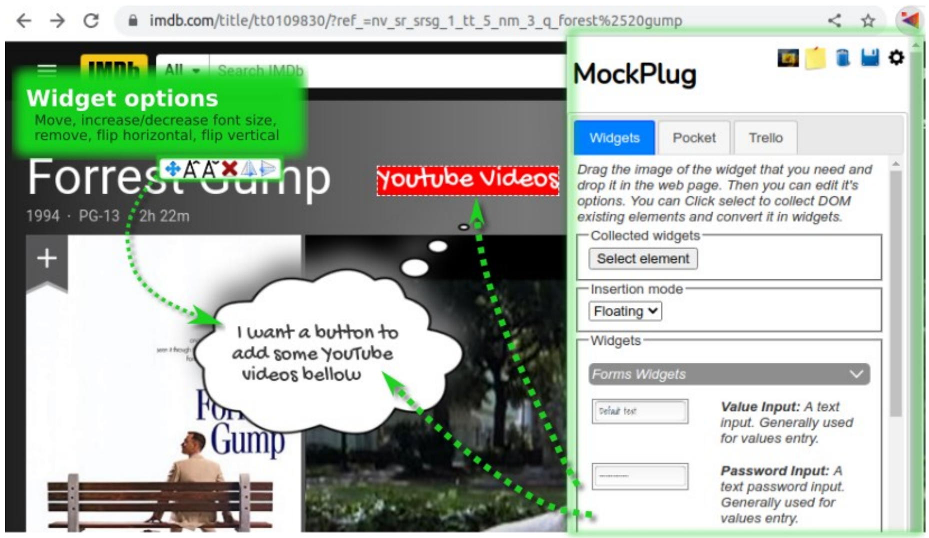


Figure 2. Mockplug class diagram.

3.2. Software functionalities

Figure 3 displays the Mockplug main menu, providing the end user with the ability to drag and drop elements onto the website to define a high-fidelity mockup. In this example, the user has added a red button labeled 'YouTube Videos' and a thought annotation to clarify their requirement. Each added element features its own menu of options, allowing the user to perform various actions related to it. In **Figure 3**, the menu options for the thought annotation are highlighted, enabling the user to move it, adjust font size, remove it, or flip it horizontally or vertically.


imdb.com/title/tt0109830/?ref_=nv_sr_srsg_1_tt_5_nm_3_q_forest%2520gump
Widget options
Move, increase/decrease font size, remove, flip horizontal, flip vertical
Forrest Gump
1994 · PG-13 · 2h 22m
Youtube Videos
I want a button to add some YouTube videos bellow
MockPlug
Widgets
Pocket
Trello
Select element
Floating
Forms Widgets

Drag the image of the widget that you need and drop it in the web page. Then you can edit it's options. You can Click select to collect DOM existing elements and convert it in widgets.

Collected widgets

Insertion mode: Floating

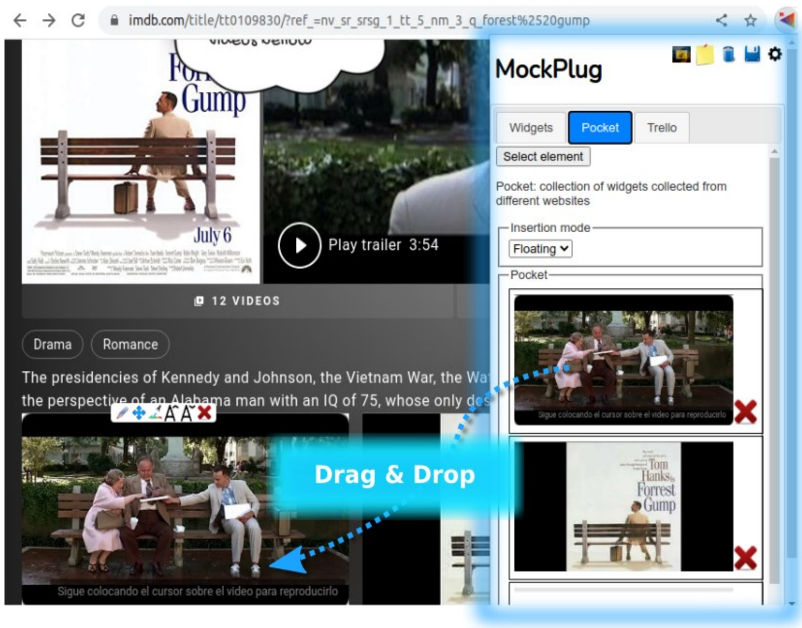
Widgets

Value Input: A text input. Generally used for values entry.

Password Input: A text password input. Generally used for values entry.

Figure 3. Mockplug main menu.

In addition to buttons and thought bubbles, users can also add form, content, and annotation widgets from the main menu. These include text inputs, passwords, selections, lists, paragraphs, images, and post-its, among others. Furthermore, users have the option to select existing elements within the web application and incorporate them into the mockup. Additionally, users can gather new elements from any location on the web for later use in the mockup. In **Figure 4**, for example, videos were collected from YouTube.


imdb.com/title/tt0109830/?ref_=nv_sr_srsg_1_tt_5_nm_3_q_forest%2520gump
Play trailer 3:54
12 VIDEOS
Drama
Romance
Drag & Drop
MockPlug
Widgets
Pocket
Trello
Select element
Floating
Sigue colocando el cursor sobre el video para reproducirlo

The presidencies of Kennedy and Johnson, the Vietnam War, the Wa… the perspective of an Alabama man with an IQ of 75, whose only d…

Pocket: collection of widgets collected from different websites

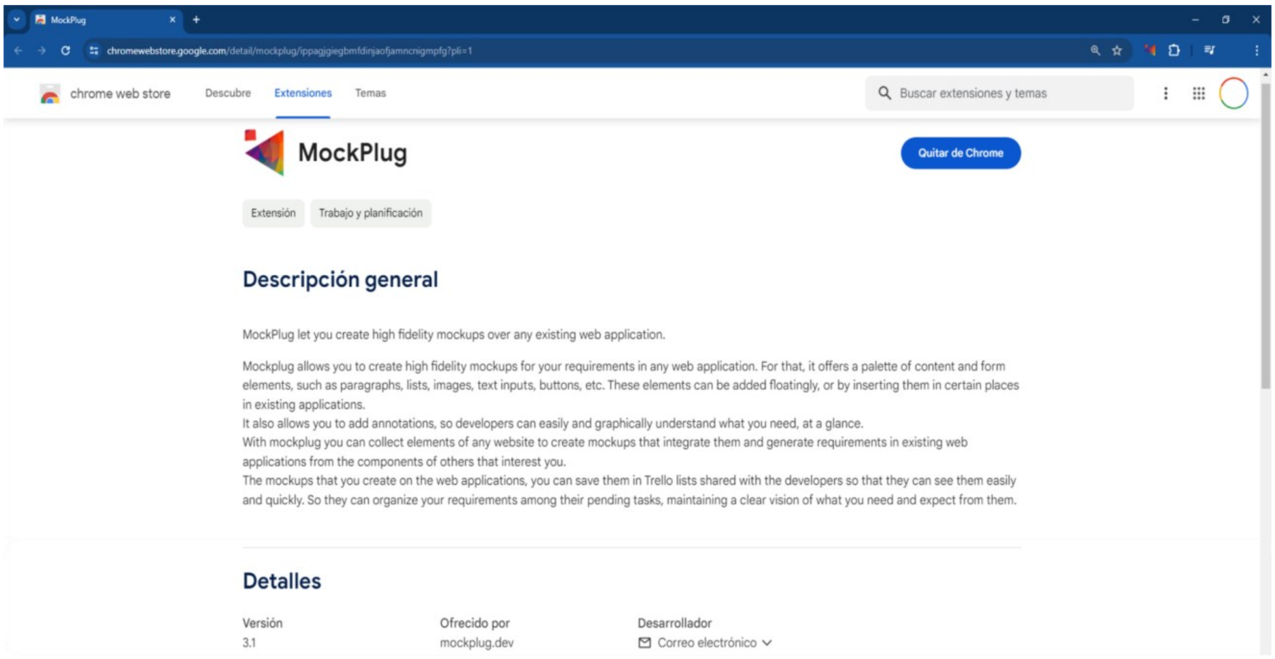
Insertion mode: Floating

Pocket

Figure 4. Mockplug pocket menu.

3.3. Software distribution

The Chrome Web Store is the official online store for users of the Google Chrome browser. By way of contextualization, in February 2024, Google Chrome absorbed more than 64% of the browser market [15,16]. Considering that there are approximately 5.3 billion monthly active Internet users, an estimated 3.46 billion people worldwide use it. In the Chrome Web Store, users can choose themes, extensions and apps to customize their browsing experience. Mockplug is distributed through this official repository. **Figure 5** shows the front page of the extension's entry in the Chrome Web Store (https://bit.ly/mockplug).


MockPlug
chromewebstore.google.com/detail/mockplug/ippagjgiegbmfdinjaofjamncnigmpfg?pli=1
chrome web store
Descubre
Extensiones
Temas
Buscar extensiones y temas
MockPlug
Quitar de Chrome
Extensión
Trabajo y planificación

Descripción general

MockPlug let you create high fidelity mockups over any existing web application.

Mockplug allows you to create high fidelity mockups for your requirements in any web application. For that, it offers a palette of content and form elements, such as paragraphs, lists, images, text inputs, buttons, etc. These elements can be added floatingly, or by inserting them in certain places in existing applications.
It also allows you to add annotations, so developers can easily and graphically understand what you need, at a glance.
With mockplug you can collect elements of any website to create mockups that integrate them and generate requirements in existing web applications from the components of others that interest you.
The mockups that you create on the web applications, you can save them in Trello lists shared with the developers so that they can see them easily and quickly. So they can organize your requirements among their pending tasks, maintaining a clear vision of what you need and expect from them.

Detalles

Versión	Ofrecido por	Desarrollador
3.1	mockplug.dev	Correo electrónico

Figure 5. Mockplug in chrome web store.

4. Illustrative examples

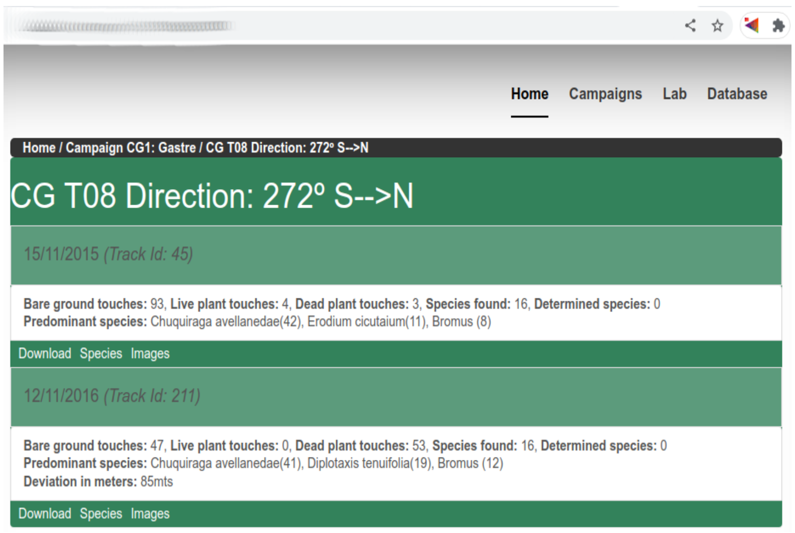
4.1. Traditional web development

The data collection process described in the following example involves scientists equipped with a mobile application for recording flora-related data directly in situ [17]. These mobile applications facilitate real-time data capture during extensive flora surveying expeditions conducted along the pathways of each campaign. Upon completion of these expeditions, scientists return to the laboratory, where the collected data is synchronized with the LeafLab web application.

Figure 6 illustrates a section of the LeafLab interface, showing a scientist reviewing the list of visits conducted along a track.

From this part of the application, the scientist can download a spreadsheet with the collected data, as well as view the photos taken during each survey, and finally access the complete list of species identified during the same. However, the user desires the ability to visualize the routes taken on a map, with each survey point represented by a red marker.

Figure 7 illustrates how, to specify their requirement, using Mockplug on the LeafLab web application accessed in the laboratory via their web browser, the user has added a map displaying the routes taken in some location. To indicate how to access the map, they have added a link titled 'map' in each of the previously described menu options. In this way, they have added five elements: two links, a representative map figure, and also clarified their needs by incorporating two annotations, one in the form of a thought bubble and the other in the form of a post-it note.



Home / Campaign CG1: Gastre / CG T08 Direction: 272º S-->N

CG T08 Direction: 272º S-->N

15/11/2015 *(Track Id: 45)*

Bare ground touches: 93, **Live plant touches:** 4, **Dead plant touches:** 3, **Species found:** 16, **Determined species:** 0
Predominant species: Chuquiraga avellanedae(42), Erodium cicutaium(11), Bromus (8)

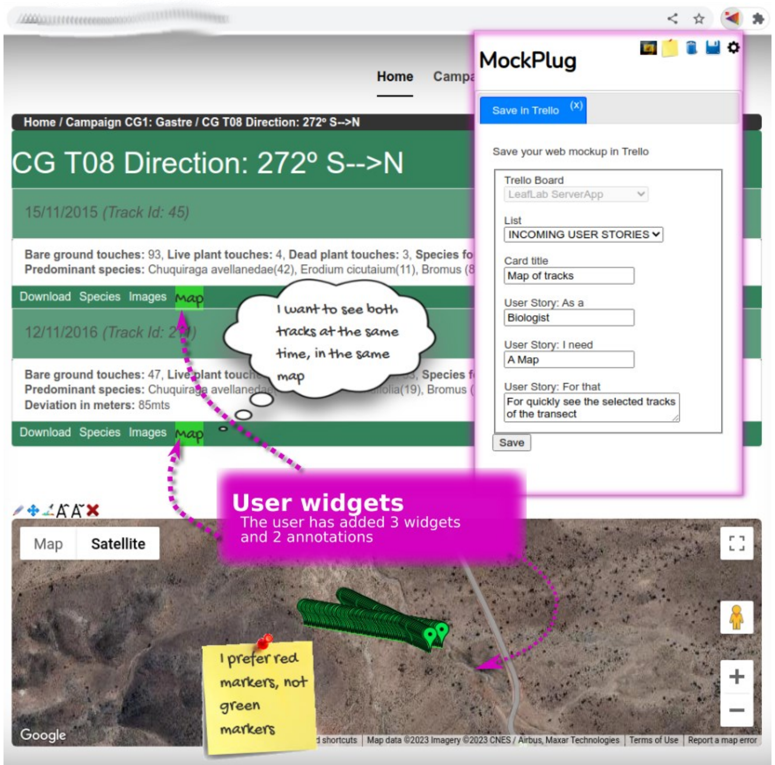
Download Species Images

12/11/2016 *(Track Id: 211)*

Bare ground touches: 47, **Live plant touches:** 0, **Dead plant touches:** 53, **Species found:** 16, **Determined species:** 0
Predominant species: Chuquiraga avellanedae(41), Diplotaxis tenuifolia(19), Bromus (12)
Deviation in meters: 85mts

Download Species Images

Figure 6. LeafLab web application.


Home
Campa
Download Species Images Map
Map
Satellite
Google
I want to see both tracks at the same time, in the same map
I prefer red markers, not green markers
User widgets
The user has added 3 widgets and 2 annotations
Map data ©2023 Imagery ©2023 CNES / Airbus, Maxar Technologies | Terms of Use | Report a map error

MockPlug

Save in Trello (x)

Save your web mockup in Trello

Trello Board: LeafLab ServerApp
List: INCOMING USER STORIES
Card title: Map of tracks
User Story: As a: Biologist
User Story: I need: A Map
User Story: For that: For quickly see the selected tracks of the transect

Save

Figure 7. Mockplug modeling over leaf lab web application example.

Then, as depicted in the same **Figure 7**, the user communicates their requirement by completing a form where they provide a title for their need and accompany it with a user story. As illustrated in **Figure 8**, this user story is automatically listed on the Kanban boards of the Leaf Lab application developers. The user story includes an image of the mockup model created by the user, and once it is on a Trello card, developers can understand the user requirements at a glance.

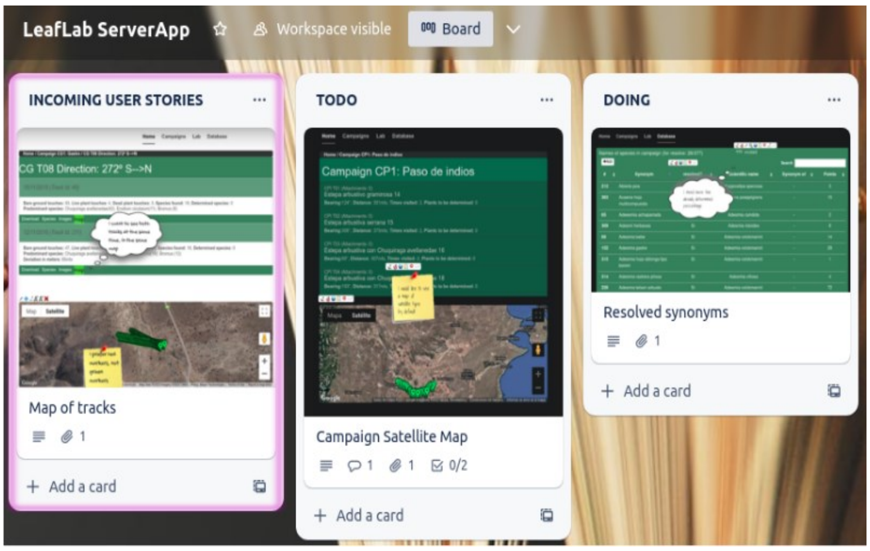
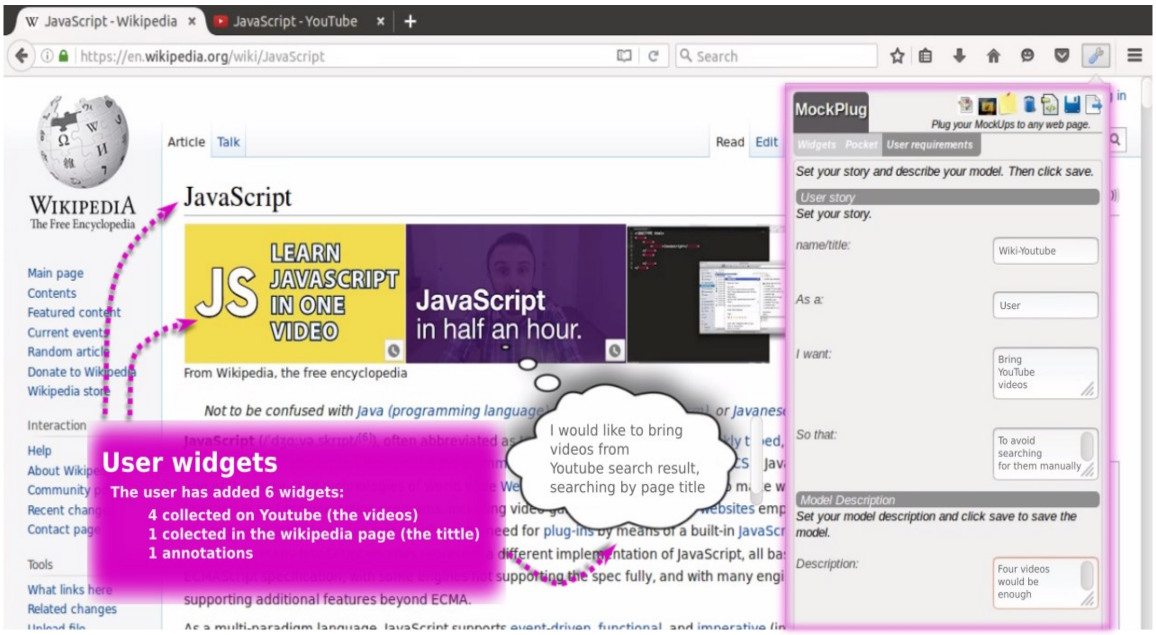


Figure 8. Mockplug model on Trello.

4.2. Web augmentation development

In this example, we illustrate how Mockplug enables end users to create web augmentation artifacts based on defined requirements. As depicted in **Figure 9**, a Wikipedia user has extracted elements from YouTube search results corresponding to the title of the Wikipedia page they were reading. Specifically, the requirement dictates that upon browsing a Wikipedia page, the first four videos from the YouTube search results are automatically embedded.


JavaScript - Wikipedia
JavaScript - YouTube
https://en.wikipedia.org/wiki/JavaScript
JavaScript
MockPlug
Plug your MockUps to any web page.
Widgets
Pocket
User requirements
I would like to bring videos from Youtube search result, searching by page title

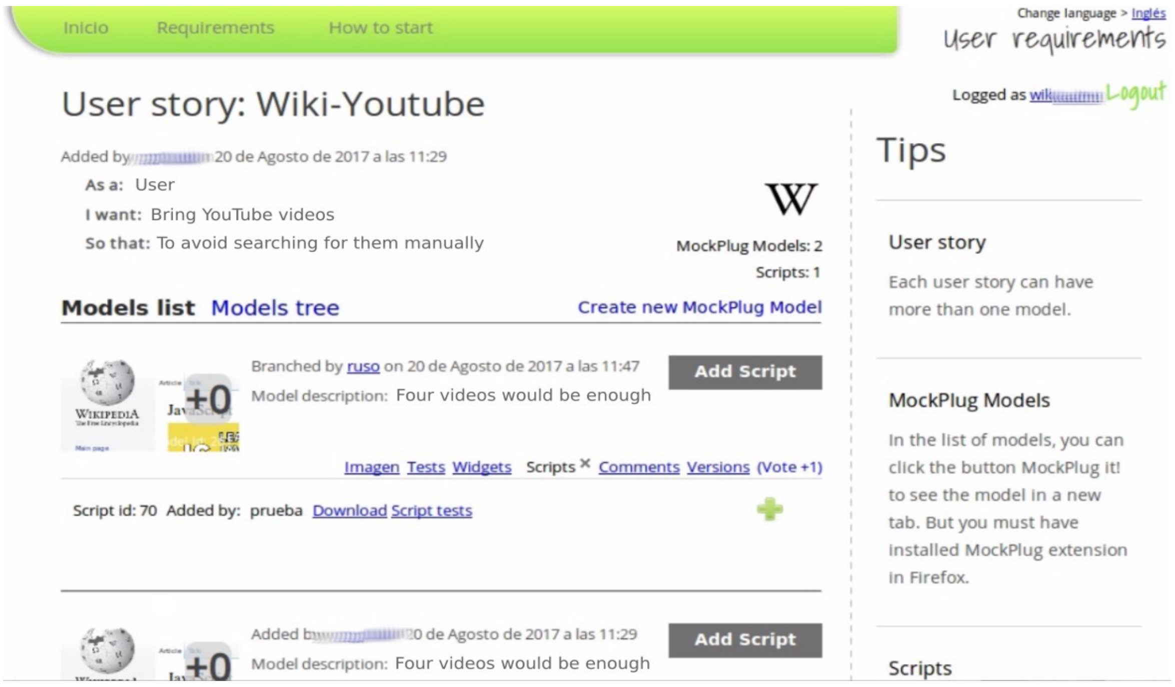
User widgets
The user has added 6 widgets:
4 collected on Youtube (the videos)
1 colected in the wikipedia page (the tittle)
1 annotations

Set your story and describe your model. Then click save.
User story
Set your story.
name/title: Wiki-Youtube
As a: User
I want: Bring YouTube videos
So that: To avoid searching for them manually
Model Description
Set your model description and click save to save the model.
Description: Four videos would be enough

Figure 9. Mockplug model version for Wikitube artifact.

This requirement is indeed authentic, as evidenced by the existence of the Wikitube augmentation artifact shared by a member of the GreasyFork augmentation artifacts community (https://greasyfork.org/es/scripts/12423-wikitube-youtube-on-wikipedia-wikiwand). In the case inspiring this example, the artifact was created and shared by the end user drhouse. This community, along with the fundamental characteristics of the artifacts developed and shared, has previously been extensively studied [9].

In the Crowdmock approach [8], augmentation artifacts are developed from Mockplug mockup models presented as user stories. These models are then stored and shared on a platform accessible to end users, both with and without programming skills. Here, users can discover requirements, refine mockups, prioritize implementation, and participate in ongoing maintenance efforts. As depicted in **Figure 10**, users can open the model in Mockplug to make relevant modifications to the high-fidelity version, collaborate on its refinement, and generate new iterations.


Inicio Requirements How to start

Change language > Inglés

User requirements

Logged as wiki[redacted] Logout

User story: Wiki-Youtube

Added by [redacted] 20 de Agosto de 2017 a las 11:29

As a: User

I want: Bring YouTube videos

So that: To avoid searching for them manually

MockPlug Models: 2

Scripts: 1

Models list Models tree Create new MockPlug Model

Branched by ruso on 20 de Agosto de 2017 a las 11:47 Add Script

Model description: Four videos would be enough

Imagen Tests Widgets Scripts × Comments Versions (Vote +1)

Script id: 70 Added by: prueba Download Script tests

Added by [redacted] 20 de Agosto de 2017 a las 11:29 Add Script

Model description: Four videos would be enough

Tips

User story

Each user story can have more than one model.

MockPlug Models

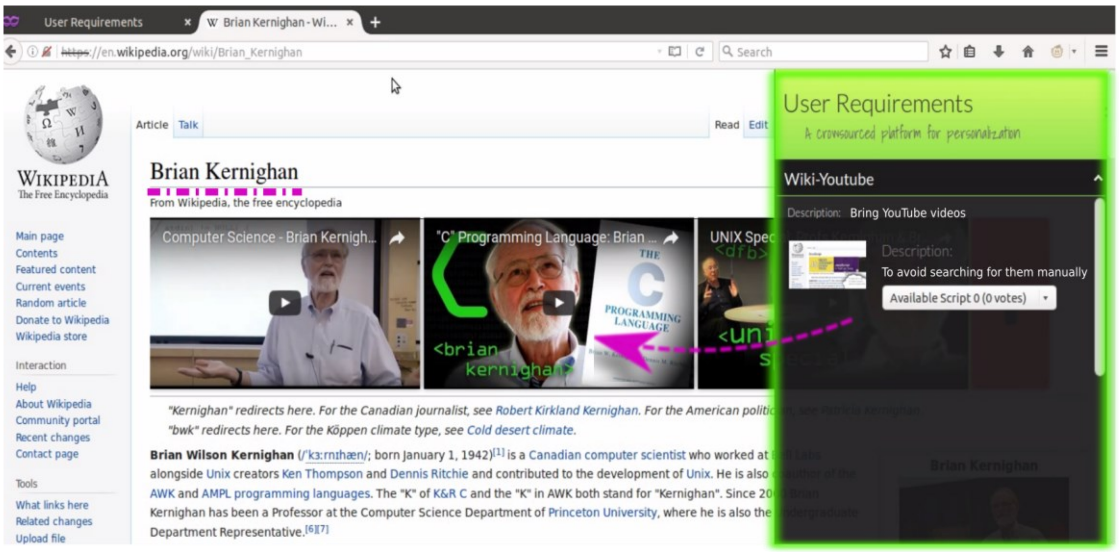
In the list of models, you can click the button MockPlug it! to see the model in a new tab. But you must have installed MockPlug extension in Firefox.

Scripts

Figure 10. Wiki-Youtube user story in user-requirements Crowdmock platform.

After an artifact is constructed in response to a requirement from an end user, it is distributed across platforms that adhere to the negotiated approach, such as the one proposed by Firmenich et al. [18]. Users of the original website can then automatically discover artifacts shared by the community and endorsed by the web application owners, as illustrated in **Figure 10**.

Figure 11 illustrates the dissemination of requirements through the experimental user-requirements crowdsourcing platform for the implementation of augmentation requirements. Finally, **Figure 12** demonstrates how users can participate in maintaining DEOI (Dynamic Element of Interest) references. These references may fail when the artifacts typically reference elements of the DOM tree of web applications, and changes made by the application owners can cause the entire augmentation artifact to fail. Users can then collect new DEOI references for needed elements.


User Requirements
A crowsourced platform for personalization
Wiki-Youtube
Description: Bring YouTube videos
Description:
To avoid searching for them manually
Available Script 0 (0 votes)
Brian Kernighan
From Wikipedia, the free encyclopedia
Computer Science - Brian Kernigh...
"C" Programming Language: Brian ...
UNIX Spec

"Kernighan" redirects here. For the Canadian journalist, see Robert Kirkland Kernighan. For the American politic

"bwk" redirects here. For the Köppen climate type, see Cold desert climate.

Brian Wilson Kernighan (/ˈkɜːrnɪhæn/; born January 1, 1942)[1] is a Canadian computer scientist who worked at

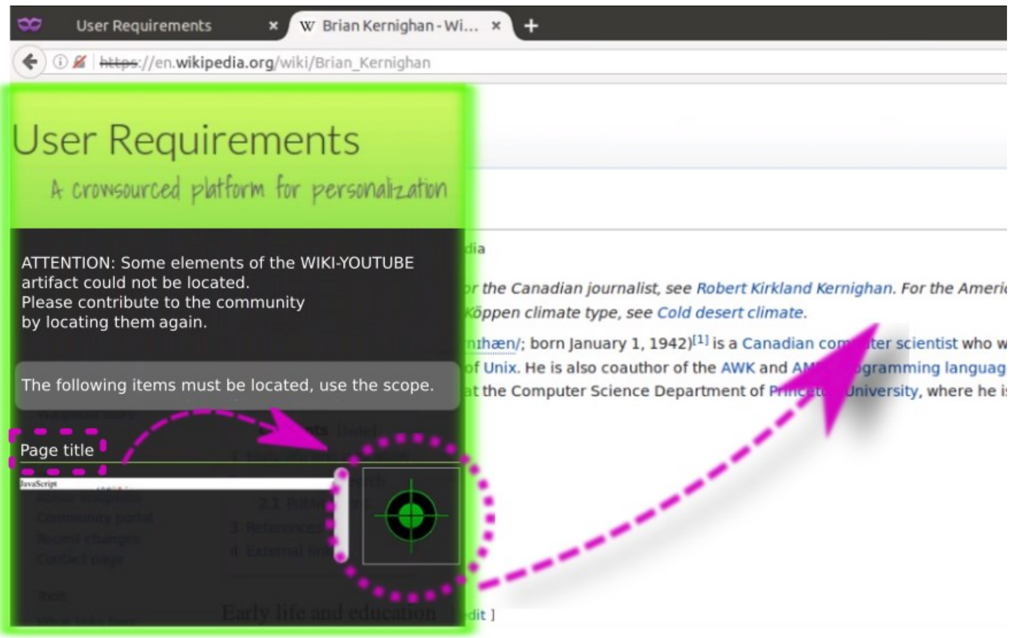
alongside Unix creators Ken Thompson and Dennis Ritchie and contributed to the development of Unix. He is also

AWK and AMPL programming languages. The "K" of K&R C and the "K" in AWK both stand for "Kernighan". Since 20

Kernighan has been a Professor at the Computer Science Department of Princeton University, where he is also the

Department Representative.[6][7]

Figure 11. Example web augmentation artifacts available for Wikipedia.



ATTENTION: Some elements of the WIKI-YOUTUBE artifact could not be located.
Please contribute to the community
by locating them again.

The following items must be located, use the scope.

Figure 12. DEOI maintenance menu, prompting the user to relocate the element "Page title" of the artifact.

All videos related to this example can be viewed in Youtube (https://bit.ly/mockplug-youtube).

5. Impact discussion

The experiments conducted with the Mockplug tool have demonstrated that users significantly reduce the time required to define their requirements, as well as achieve better results [19], as shown in **Figure 13**.

It has been proven that Mockplug, integrated into a crowdsourcing platform [8], facilitates end-users in obtaining web augmentation software artifacts by themselves. Its integration has also been tested in a negotiation approach between end-users developing augmentation artifacts and website owners. In this approach, website owners take responsibility for endorsing the deployment of artifacts of interest to all users on their sites, through the aforementioned platform [18].

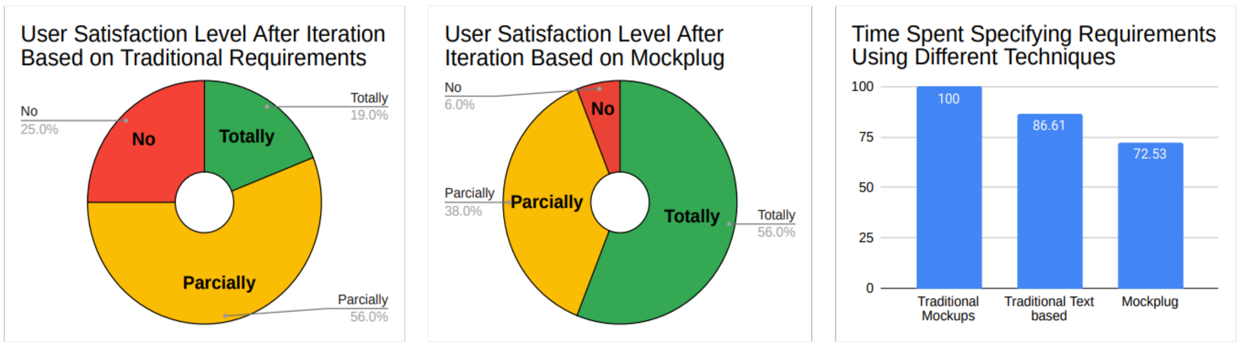


Figure 13. Improvement in perceived satisfaction and time reduction with Mockplug.

On the other hand, when Mockplug was integrated into the traditional agile development process in existing web applications through Kanban boards, it was proven that developers in their environments could retrieve the files involved in the implementation of the requirement in almost all cases [7].

Until now, Mockplug has been maintained as a tool to provide experimental support to the mentioned approaches, and its use has been limited to academic environments where the underlying topics were addressed. This includes students in undergraduate and postgraduate courses, as well as users in general invited to experiments. An estimated 200 occasional users have utilized the tool.

6. Conclusions

Mockplug opens up the possibility of engaging the end user in a completely different way than usual in the web software development process. Unexplored forms of collaboration become feasible because this tool enables quick impact on a Kanban board or even on a platform for the development of artifacts by end-users, facilitating the collection of references to DEOI (DOM Element of interest) and the detection of software components relevant to requirement implementation.

In the future, this software tool may be extended to support end-users with NLP (Natural Language Processing) techniques to explore the potential of AI in defining mockups and requirements, and integration into Kanban boards. The experiments conducted with the tool have demonstrated that users significantly reduce the time required to define their requirements, as well as achieve better results. On the other hand, it has been proven that developers in their environments can retrieve the files involved in the implementation of the requirement in almost all cases.

Author contributions: Conceptualization, DF; methodology, DF; software, DF, GM and NC; validation, DF, GM and NC; formal analysis, DF, LM, GM and NC; investigation, DF, LM, GM and NC; resources, DF, LM, GM and NC; data curation, DF, LM, GM and NC; writing—original draft preparation, DF, LM, GM and NC; writing—review and editing, DF, LM, GM and NC; visualization, DF, LM, GM and NC; supervision, DF and LM; project administration, DF and LM; funding acquisition, DF and LM. All authors have read and agreed to the published version

of the manuscript.

Conflict of interest: The authors declare no conflict of interest.

References

1. Macaulay L. Requirements for requirements engineering techniques. In: Proceedings of the Second International Conference on Requirements Engineering; 1996. pp. 157-164.
2. Beynon-Davies P, Holmes S. Integrating rapid application development and participatory design. IEE Proceedings-Software. 1988; 145(4): 105-112. doi: 10.1049/ip-sen:19982196
3. Moore JM. Communicating requirements using end-user gui constructions with argumentation. In: Proceedings of the 18th IEEE International Conference on Automated Software Engineering, 2003. pp. 360-363
4. Rashid A, Meder D, Wiesenberger J, et al. Visual Requirement Specification in End-User Participation. In: Proceedings of the 2006 First International Workshop on Multimedia Requirements Engineering (MERE'06-RE'06 Workshop). doi: 10.1109/mere.2006.7
5. Rivero JM, Grigera J, Rossi G, et al. Mockup-Driven Development: Providing agile support for Model-Driven Web Engineering. Information and Software Technology. 2014; 56(6): 670-687. doi: 10.1016/j.infsof.2014.01.011
6. Molina-Ríos J, Pedreira-Souto N. Comparison of development methodologies in web applications. Information and Software Technology. 2020; 119: 106238. doi: 10.1016/j.infsof.2019.106238
7. Marticorena LG, Morales LA, Antonelli L, et al. Development iterations based on web augmentation and context tasks. Multimedia Tools and Applications. 2022; 82(8): 11793-11817. doi: 10.1007/s11042-022-13694-2
8. Firmenich D, Firmenich S, Rivero JM, et al. CrowdMock: an approach for defining and evolving web augmentation requirements. Requirements Engineering. 2016; 23(1): 33-61. doi: 10.1007/s00766-016-0257-3
9. Firmenich D, Firmenich S, Rossi G, et al. Engineering Web Augmentation software: A development method for enabling end-user maintenance. Information and Software Technology. 2022; 141: 106735. doi: 10.1016/j.infsof.2021.106735
10. Díaz O. Understanding web augmentation. In: Current Trends in Web Engineering: Proceedings of the ICWE 2012 International Workshops: MDWE, ComposableWeb, WeRE, QWE, and Doctoral Consortium. Springer Berlin Heidelberg; 2012.
11. Aldalur I, Winckler M, Díaz O, Palanque P. Web Augmentation as a Promising Technology for End User Development. In: New Perspectives in End-User Development. Springer, Cham; 2017.
12. Aldalur I. Web Augmentation: A systematic mapping study. Science of Computer Programming. 2024; 232: 103045. doi: 10.1016/j.scico.2023.103045
13. Ko AJ, Abraham R, Beckwith L, et al. The state of the art in end-user software engineering. ACM Computing Surveys. 2011; 43(3): 1-44. doi: 10.1145/1922649.1922658
14. Urbieta M, Mahl F, Rossi G, et al. A Web-based Model-driven Platform for Web Augmentation. In: Proceedings of the 15th International Conference on Web Information Systems and Technologies; 2019. doi: 10.5220/0008559300002366
15. Browser Market Share of 2024 (Different Regions & Devices). Available online: https://www.yaguara.co/browser-market-share/ (accessed on 22 May 2024).
16. Desktop Browser Market Share Worldwide—April 2024. Available online: https://gs.statcounter.com/browser-market-share/desktop/worldwide/#monthly-202212-202312 (accessed on 22 May 2024).
17. Almonacid S, Klagges MR, Navarro P, et al. Mobile and wearable computing in patagonian wilderness. In: Cloud Computing and Big Data. JCC&BD 2019. Communications in Computer and Information Science. Springer, Cham; 2019. pp. 137-154.
18. Firmenich D, Firmenich S, Rossi G, et al. User interface adaptation using web augmentation techniques: towards a negotiated approach. In: Engineering the Web in the Big Data Era. ICWE 2015. Lecture Notes in Computer Science. Springer, Cham.; 2015. pp. 147-164.
19. Firmenich D, Firmenich S, Rivero JM, et al. A platform for web augmentation requirements specification. Web Engineering: In: Web Engineering. ICWE 2014. Lecture Notes in Computer Science. Springer; 2014. pp. 1-20.