informatics

MDPI

Article

How to Improve Usability in Open-Source Software Projects? An Analysis of Evaluation Techniques Through a Multiple Case Study

Lucrecia Llerena [1], John W. Castro [2,*], Rosa Llerena [3] and Nancy Rodríguez [1]

[1] Software Engineering Program, Faculty of Engineering Sciences, Quevedo State University, Quevedo 120301, Ecuador; lllerena@uteq.edu.ec (L.L.); nrodriguez@uteq.edu.ec (N.R.)
[2] Departamento de Ingeniería Informática y Ciencias de la Computación, Facultad de Ingeniería, Universidad de Atacama, Copiapó 1532297, Chile
[3] Economics Program, Faculty of Economics and Financial Social Sciences, Quevedo State University, Quevedo 120301, Ecuador; rllerenag@uteq.edu.ec
* Correspondence: john.castro@uda.cl

Abstract: Open-source software has experienced steady growth, driving increased research. However, open-source communities still need standardized processes to ensure software quality, and their characteristics make it challenging to adopt many usability techniques from human–computer interaction directly. Our study aims to adapt and evaluate the feasibility of three usability evaluation techniques—cognitive walkthrough, formal usability testing, and thinking aloud—across three open-source projects (Code::Blocks, Freeplane, and Scribus) from the development team's perspective. We participated as volunteers in these projects, employing a multiple case study method. We found that usability techniques were not systematically adopted, and procedures specific to open-source projects were lacking. We also identified adverse conditions, such as limited user participation, that hindered adoption. We proposed adaptations to each technique and formalized procedures to address these challenges and apply them in open-source contexts. Additionally, we developed a framework for integrating usability evaluation into OSS projects. To address this, we detailed our framework's phases, tasks, and artifacts to ensure reusability and adaptability across OSS contexts, providing practical steps for implementation and future validations. In conclusion, usability techniques must be adapted for open-source software, considering the projects' unique characteristics and philosophy. Although obstacles exist, such as user participation, applying adapted usability techniques in open-source projects is feasible.

Keywords: cognitive walkthrough; formal usability testing; open-source software; thinking aloud; usability techniques

Academic Editor: Roberto Theron

Received: 30 September 2024
Revised: 13 December 2024
Accepted: 26 December 2024
Published: 27 January 2025

Citation: Llerena, L.; Castro, J.W.; Llerena, R.; Rodríguez, N. How to Improve Usability in Open-Source Software Projects? An Analysis of Evaluation Techniques Through a Multiple Case Study. *Informatics* **2025**, *12*, 12. https://doi.org/10.3390/informatics12010012

1. Introduction

Open-source software (OSS) represents a distribution model (source code) that guarantees free access, allowing the user to install the software without an additional purchase and to access its source code if they want to contribute to its development. This type of software allows those qualified to test and fix bugs without help instead of waiting for someone else to do it [1]. OSS communities still need to establish standard processes to ensure that the product they develop meets quality attributes [2]. Due to the significant increase in the number of non-developer users of OSS applications, there is a growing interest in developing usable OSS [3,4]. Moreover, usability is an essential characteristic,

and it is considered a quality attribute that evaluates how easy it is to use a graphical user interface [5–8]. According to the human–computer interaction (HCI) approach, there is an excellent variety of usability techniques. This variety is evidenced in the catalog compiled by Ferré et al. [9], which contains 55 HCI techniques for integration into the software engineering development process. Several authors have recognized low usability in OSS projects [3,4,10,11]. Its incorporation into the OSS development process is complex due to the unusual characteristics of these communities. The main challenges faced by these projects in terms of usability include (i) a culture focused on functionality development, (ii) the global geographic location of its members, (iii) scarcity of resources, and (iv) a culture that may be somewhat alien to interaction design. This prevents many HCI usability techniques from being adopted directly [12].

Our research determines how to incorporate three usability techniques (cognitive walkthrough, formal usability testing, and thinking aloud) into the development process of three OSS projects (Code::Blocks, Freeplane, and Scribus). To do so, we previously analyzed and identified the impediments that must be solved to apply these three techniques in the three OSS projects, considering their specific characteristics. Although there is a framework for integrating techniques in OSS developments [12], this framework is general. It does not detail the steps and artifacts that should be used to adapt the techniques. The solutions to these impediments correspond to the adaptations that must be made to the steps of the methods to allow their incorporation into the OSS development process. In other words, with our research, we want to improve usability in OSS projects, making their developers recognize the value and benefits of applying evaluation techniques. For this, it is necessary to evaluate such OSS projects by adopting HCI methods that allow usability problems to be identified for their subsequent solution.

Nielsen [13] states that usability is not a single, one-dimensional property of a user interface but comprises multiple components, such as learnability, efficiency, productivity, errors, and satisfaction. Based on this definition, we can see usability as a software evaluation metric oriented to the components to meet a quality standard in software systems. To meet this quality, it is necessary to understand the definition of low usability in a project [14], which can be determined by little or no user interaction with the system. In addition, usability in non-commercial software systems should be more recognized [14]. Therefore, the need arises to evaluate OSS projects by applying different techniques, among which stand out cognitive walkthrough, formal usability testing, and thinking aloud. Integrating techniques related to usability evaluation in OSS projects is a complicated process that has yet to be investigated in depth [15–19]. Existing studies suggest that the techniques should be reconceptualized. However, they need to detail how to adjust the methods. To the best of our knowledge, only the studies by Nichols and Twidale [20] and Ternauciuc and Vasiu [21] propose some general ideas for improving the usability of OSS projects. Although there are studies on usability in OSS, there is no standardized procedure for its incorporation in OSS developments. Only the study by Castro [12] analyzes the usability problems and techniques used in OSS projects in an integrated way, proposing a general framework for integrating these techniques and considering the characteristics and philosophy of the OSS community. In the literature, some works have reported the use of cognitive walkthrough [22–26], formal usability testing [27–29], and thinking aloud [2,16,23,30–32]. However, these works only report the results of applying usability evaluation techniques and do not specify the adaptations used in OSS projects.

The current literature highlights critical challenges in usability evaluation in OSS projects, including the lack of systematic comparative studies and specific adaptations of techniques such as cognitive walkthrough, formal usability testing, and thinking aloud for these environments. Our research addresses these gaps by providing a detailed and

practical framework that integrates these techniques, adapted to the unique characteristics of OSS projects, such as their distributed development and lack of usability experts. In addition, we evaluate usability from the perspective of end users and developers, providing a complete view. Finally, we use qualitative and quantitative methods for a more detailed evaluation.

Cognitive walkthrough (CW) [33] is used to identify usability problems in interactive systems. It aims to know how easily new users can perform tasks with the software system. This method is one of the most appreciated due to its speed in generating results at low cost. Formal usability testing (FUT) [34] corresponds to the most common tests in the HCI field. In these tests, the user is asked to perform a series of tasks with the system prototype to observe what problems the user experiences, what mistakes they make, and how they recover from them. Thinking aloud (TA) is very effective when it is required to obtain flaws in software development [16,30]. In this technique, it is necessary to carefully observe what the user says and does and how they behave in front of a software system. Starting from their emotions, we can find usability improvements that would not be obtained using traditional methods (e.g., surveys). In addition, TA allows data to be received from a user through a set of activities to be tested on a system, verbalizing their thoughts. Users allow an observer to determine what they are doing with the interface and why they are doing it [16]. CW is fast and inexpensive for identifying usability problems; FUT evaluates effectiveness, efficiency, and user satisfaction; and TA provides a deep understanding of user interactions and thoughts while using the system.

To choose the projects for this research, we used SourceForge, one of the most popular OSS repositories [35]. For the selection of OSS projects, we have considered the following criteria: (i) availability for the Windows operating system; (ii) tool related to education and engineering; (iii) in a beta or stable state; (iv) constant updates; (v) active user community; (vi) developed in a programming language known to researchers; (vii) projects in early stages of development, where essential functionalities are still being added; (viii) presence of a critical mass of users to determine the actual user segments of an application; and (ix) frequency of activity and popularity score of the application as rated by its users. Considering the above, we selected the Code::Blocks, Freeplane, and Scribus projects. These projects constitute a suitable framework for our study because specific authors claim that OSS communities generally do not know about usability techniques [21,36,37], do not have resources for usability testing, and usability experts are not involved in these projects [20,21,36,38–40]. Some OSS projects need more knowledge about the techniques available to improve usability [15,40–43].

Code::Blocks is a cross-platform IDE for programming in the C++ and C languages. It is designed to be extensible and fully configurable. In the Code::Blocks forum, users comment on difficulties encountered and criticisms of the user interface (e.g., lack of a default path to save a project). Freeplane allows the creation of mind maps. In the Freeplane forum, users comment on some difficulties encountered and criticize the user interface (e.g., the location of some options in the menus could be more evident). Scribus allows the layout of web pages, typesetting, and preparing professional quality image files; it can adapt to any language and user. Scribus users also found some difficulties and criticisms of the user interface, such as (i) unintuitive interface; (ii) it is not possible to change the font size immediately; (iii) it is not possible to insert an image directly from the web into the project; (iv) when trying to change the opacity, it is not possible; (v) when selecting the edit image option an error is displayed indicating that the function does not exist; and (vi) the icons in the properties and header window are not very intuitive.

In order to solve the difficulties encountered by users, we propose the adaptation of CW, FUT, and TA to the context of OSS projects. To adapt these techniques, we will start

with the integration framework proposed by Castro [12]. In addition, for the application of CW, FUT, and TA in OSS projects, several artifacts were designed, such as (i) a document with the description of the tasks to be performed by the users, (ii) a document to record user information by the evaluator, (iii) a document to record the time spent by each user in performing the tasks, (iv) a document for the user to record their experience and observations, (v) a letter to request authorization from the developer, (vi) consent for image capture and video recording, and (vii) a formal invitation to users, sent by e-mail, to participate in the usability evaluation. These artifacts will be tools to apply CW, FUT, and TA.

As mentioned above, the three OSS projects were not designed with usability aspects in mind. Instead, they were built mainly while thinking about the functionalities proposed by their developers. Therefore, to adapt these three techniques, we consider the typical scenario of OSS projects, i.e., (i) volunteers work remotely, and (ii) they usually do not have a usability expert to design the evaluation experience nor perform the final analysis of the improvement proposals. Therefore, our proposal is novel in that it submits several methods adapted to a complex application scenario, such as that of OSS communities. Furthermore, considering that techniques in the HCI area, in general, do not have formalized procedures for their application in this type of OSS development, it is necessary to standardize these practices in the OSS context, thus ensuring usability improvements that are replicable and sustainable in the long term [12].

The present work also makes a significant contribution to the field of Software Engineering, in particular in OSS developments because, as mentioned above, to our knowledge, there are no works that report how to incorporate CW, FUT, and TA in OSS projects, specifying the adaptations to be made and formalizing the application process of each of these techniques. Thus, in particular, our research presents the following contributions: (i) We identify the adverse conditions that hinder the application of CW, FUT, and TA in OSS developments. (ii) We propose adaptations for each of the steps of CW, FUT, and TA based on the identified adverse conditions that allow their incorporation into the OSS development process. (iii) We validate the proposed adjustments by applying CW, FUT, and TA to real OSS projects. (iv) We suggest recommendations to improve the user interface of Code::Blocks, Freeplane, and Scribus through the identification of usability issues as a result of applying the CW, FUT, and TA. (v) Based on our experience participating as volunteers in different OSS projects, we propose a framework to integrate techniques related to usability evaluation in this type of OSS project.

This article is organized as follows. In Section 2, we describe related work. In Section 3, we report the research method. Section 4 describes the identified adverse conditions and usability technique adaptations. In Section 5, we present the design of the multiple case study. In Section 6, we report the results of the multiple case study. Section 7 presents a framework for usability evaluation in the OSS projects. In Section 8, we discuss the results. In Section 9, we describe this study's limitations. Finally, in Section 10, we present conclusions and future research.

2. Related Work

This section reports the work related to applying CW, FUT, and TA in OSS projects. These techniques have been widely recognized in the HCI area for their ability to evaluate and improve the user experience in software systems. The description of the work will serve as the basis for the proposed adaptations to the usability techniques in the context of OSS development, presented in later sections of this paper.

Few studies have recently reported using the CW in OSS developments [22–26]. In the study by Assal et al. [22], the authors use the CW to evaluate a system called Caesar.

The authors combined a cognitive dimension framework with the CW in the usability evaluation. The authors consider the application interface with the participants (developers) performing different previously defined tasks. In the study by Ledesma et al. [23], the authors report the application of the CW for evaluating the medical software library hFigures, with the participation of three experts. A CW is performed with several end users to assess the usability of BiDaML [24]. These users highlighted that BiDaML was easy to understand and learn. The authors obtained favorable results. Weninger et al. [25] conducted a study to evaluate the AntTracks Analyzer tool using the CW. The main objective of the CW was to reveal and correct possible usability flaws. In the study by Tegos et al. [26], the authors evaluated the usability of the PeerTalk system. The study involved four MOOC instructors and two experts. In this study, the CW technique was used to assess the usability of the system functions when users performed the given tasks.

To the best of our knowledge, very few research papers report applying the FUT technique in OSS projects [2,27–29]. In the study by Andreasen et al. [2], testing is mentioned in the context of usability evaluation methods, highlighting that these methods are not widely used in OSS projects due to the lack of resources and evaluation methods that fit the OSS paradigm. The study of Cemellini et al. [27] reports the implementation of a 3D cadastre prototype. The authors performed a test to obtain feedback from different groups of users. During the application of FUT, the test users were subdivided into groups according to different professional domains and expertise. The usability test result was crucial to pinpoint the limitations detected at this early prototype development stage. In Hall's study [28], the authors examine OSS's usability, focusing on why this quality attribute is often overlooked. The authors apply FUT to the GNOME project, a popular OSS desktop environment. The application of this technique allowed the authors to collect data suggesting usability features or issues. In the work of Pietrobon et al. [29], the authors describe the development of a web OSS called Duke Surgery Research Central (DSRC). The FUT was used to evaluate the usability of the DSRC. To apply this technique, the authors used an external evaluator and recruited ten users who performed the tests proposed by this evaluator. The results of the application of the FUT were positive.

Some works in the literature report use TA in OSS projects [2,16,23,30–32]. The empirical study by Andreasen et al. [2] reports that 21% of the OSS developers surveyed have used the TA. The authors mention that developers often participate in usability evaluations, although usability professionals do not always perform these activities; instead, they follow the developers' common sense [2]. The authors identified several problems related to usability evaluation in OSS communities, including a need for more priority in systematically assessing this quality attribute and a limited understanding of usability among developers. In addition, evaluation is often performed remotely and with conventional techniques such as TA. However, this approach presents challenges in accurately identifying usability issues due to the absence of facial expressions and the technical setup required to perform the evaluation remotely. TA has been adopted to develop a web browser, a desktop utility, an operating system [16], and the TrueCrypt project [30]. The study by Terry et al. [16] highlights some usability problems in OSS, such as the difficulty in implementing a holistic design due to its distributed and voluntary development, the scarcity of usability infrastructure, and a developer-centric culture that limits the integration of user experience practices. In the TrueCrypt project, users were asked to "think out loud" while performing specific tasks, especially to indicate when they started, finished, or became stuck [30]. In addition, TA has been used in the CARTON [31], hFigures [23], and ViSH Editor [32] projects. In the CARTON project [31], the authors evaluated students. Participants were encouraged to provide feedback while building an augmented reality device by following the guides provided in the application. In the study by Ledesma et al. [23], the authors report using TA

as part of a test conducted in a usability lab. In this study, participants were asked during the evaluation to perform specific tasks while commenting on what they were doing and thinking. Finally, TA was applied to assess ViSH Editor [32]. In this case, the authors used the technique with the help of a session moderator. The moderator gave the participants the tasks and asked them to think aloud while performing them.

Usability is a critical aspect in the development of OSS projects, as it directly influences the adoption and satisfaction of its users. Some studies have evaluated usability in various OSS projects [4,11], highlighting challenges and opportunities to improve user experience. However, despite these efforts, the current literature has several essential gaps. These include the lack of systematic comparative studies that analyze different techniques related to usability evaluation across multiple case studies. Although studies in the literature evaluate the usability of individual OSS projects, they need to report the necessary adaptations to adequately apply CW, FUT, and TA techniques in these developments, considering their characteristics. Moreover, these studies must formalize or systematize each step to use the methods. Our study addresses all these gaps.

In summary, after a review of the literature related to CW, FUT, and TA, the lack of studies that specifically address the necessary adaptations that allow their application in OSS projects, considering the characteristics of this type of development, stands out. As for CW, papers that report on its adaptation for incorporation in OSS developments were not found, which highlights the need for more research in this area, such as the one presented in this study. Concerning FUT, although there are some studies on its application in OSS, the lack of detailed documentation on the steps and adaptations required for its application points to the importance of continuing to explore this line of research. Finally, although TA has been used in some OSS projects, the absence of studies describing its specific adaptations for this environment emphasizes the need for further research. In conclusion, our work seeks to fill these gaps by providing a detailed and practical framework for incorporating these three evaluation techniques in OSS projects, thus opening new opportunities for improving usability in this domain.

3. Research Method

We validated our research using a qualitative study called the multiple case study [44,45], which consists of studying the phenomenon of interest in its real context. In our case, the phenomenon of interest is incorporating usability techniques with adaptations in OSS projects. In addition, subjects are observed in their real context [46]. OSS projects are a suitable framework for our study due to the heterogeneous characteristics of the members involved in their development since OSS communities are generally unfamiliar with usability methods [21,40–42]. They usually need more resources to perform usability testing and experts to develop these projects [21]. In our research, we opted for a multiple case study approach for the following reasons: (i) It allows a broad and deep comparison of the three selected usability evaluation techniques (i.e., CW, FUT, and TA) in different OSS projects, identifying common patterns and best practices. (ii) The diversity of OSS projects requires capturing this heterogeneity, ensuring a complete and detailed representation of the usability evaluation techniques. (iii) It allows a detailed exploration of each case and an integrated synthesis of the findings, facilitating the understanding of the findings. Therefore, we selected the multiple case study to validate the feasibility of our proposal to incorporate adapted usability techniques in OSS projects. The description of the multiple case study is based on the guidelines of Runeson et al. [44]. The process used to conduct this multiple case study consists of two phases: (i) data preparation and collection and (ii) data analysis and reporting. The conduct of our multiple case study can be reviewed in detail in Figure 1.

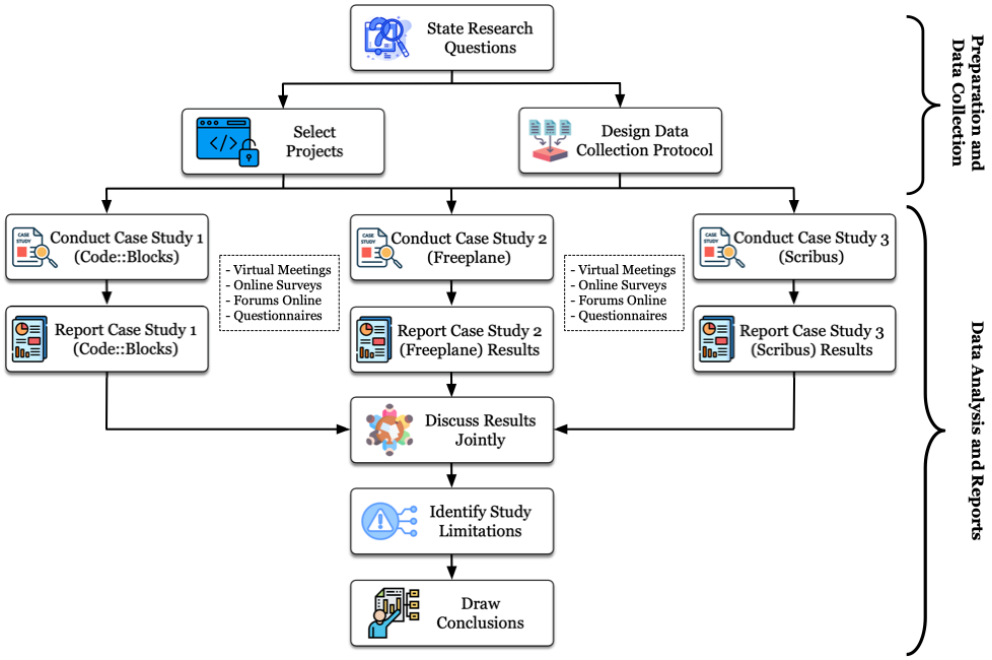

Figure 1. Activities carried out during the development of the multiple case study (adapted from [44,45]).

In the first phase, we set the research question, which focuses on incorporating techniques into the OSS development process. In the second phase, we adapt the methods applied directly to the selected OSS projects and propose incorporating improvements into each project. To ensure the reliability and validity of the data collected on usability practices in OSS projects, we employed multiple qualitative methods, including online surveys, interviews, virtual meetings, and observations. The online surveys allowed us to obtain detailed information about users' experiences and perceptions regarding the usability of the projects. The open-ended interviews provided us with an in-depth understanding of specific usability issues faced by users. In addition, virtual meetings allowed us to observe how users interacted with the OSS projects in real time, recording their immediate comments and reactions.

In addition, we involved usability experts to review our procedures and conducted pilot tests with representative users to identify potential problems and make necessary adjustments. Finally, we maintained detailed documentation of all data and methods to enable replication and independent verification of our results. These measures allowed us to obtain reliable and valid data to evaluate the proposed adaptations to the usability techniques applied to OSS projects. Our research should also mention that the three case studies are related because of their characteristics (e.g., they are OSS developments and do not involve usability experts). We also chose the multiple case study because there is little room for experimentally manipulating the phenomenon under study: the adoption of usability techniques in a real OSS project [47]. In addition, it is a qualitative study that uses techniques such as group discussion or open-ended interviews to collect data [47].

4. Identified Adverse Conditions and Usability Technique Adaptations

Unlike conventional HCI environments, OSS communities present several unique characteristics that affect the direct application of these techniques. These differences include (i) geographic dispersion of members, (ii) a focus on code development, (iii) resource constraints, and (iv) a culture that may diverge from interaction design conventions [7,12,48]. Although techniques are fundamental to improving the user experience, their implementation in OSS environments faces significant challenges. Standard procedures defined

by HCI authors rarely consider the peculiarities of OSS projects, resulting in a lack of practical guidance for their effective application in such developments. This lack extends to usability techniques, highlighting the need for a systematic approach to adapt them to the OSS context. Therefore, we propose a four-step process for adapting techniques in OSS development. First, we seek to formalize the steps necessary to adjust each usability technique, providing detailed guidance on its implementation in OSS environments. In the second stage, we conducted a thorough analysis to identify the adverse conditions that could hinder their effective implementation. With this information, in the third stage, we propose specific adaptations to overcome these obstacles and facilitate the integration of usability techniques into the OSS development process. Finally, in the fourth stage, we demonstrate the feasibility and effectiveness of these adaptations through their application in the selected OSS projects. Next, we present the development of the first three stages for adapting CW, FUT, and TA for subsequent application in the OSS projects. We develop the fourth stage in Section 5.

The three usability techniques incorporated in OSS projects share two adaptations. First, the techniques require a usability expert for their application [33,49,50], so we recommend replacing it with an HCI student under the supervision of a mentor. Rajanen and Iivari [42] studied the practical implications of involving HCI students in OSS projects with satisfactory results. Secondly, the techniques necessarily require the physical participation of users. In this case, we propose that OSS users participate remotely by performing between one and two tasks to maintain their involvement.

4.1. Cognitive Walkthrough Technique and Its Adaptations

4.1.1. Description of the Cognitive Walkthrough Technique

Among the usability inspection methods, the CW is one of the most important [34]. It is used to identify usability problems in interactive systems and aims to know how easy it is for new users to perform tasks with the software system. CW is one of the most appreciated techniques due to its speed in generating results at a low cost. According to Wharton et al. [33], the CW consists of 5 steps: (i) In step 1, users are identified, and tasks with detailed steps are designed. (ii) In step 2, the usability expert invites those who will participate in executing the CW to present the objective and the basic rules. (iii) In step 3, go through the action sequences of each task. In addition, the basic rules of CW must be enforced. (iv) In step 4, critical information is recorded. (v) In step 5, the analysts and the design team review interface improvements for troubleshooting.

4.1.2. Adaptations of the Cognitive Walkthrough Technique

The CW consists of a structured approach to evaluate the usability of a software product and has different treatments in the analyzed literature [33,34]. Although they are all similar, for processes focused on user-centered development, a suitable approach is proposed by Wharton et al. [33] because it has a simple description of the steps that facilitate its application. According to Wharton et al. [33], this technique consists of five steps. Three adaptations are necessary to incorporate CW in OSS projects. The first two were described at the beginning of Section 4. Third, prior preparation is required, e.g., conceiving the task that the user must perform to apply the technique, and in the OSS community, the work performed is entirely voluntary and developed in the free time of its members. In this case, we propose to take advantage of the tasks usually performed by the users with the OSS project.

Moreover, according to Wharton et al. [33], to execute the CW positively, at least five users must participate and be sincere in answering the usability expert's questions. In this research, we adapted the CW to facilitate its incorporation in the selected OSS project.

Table 1 summarizes the CW process. For each technique's step, we report the adverse conditions identified and the adaptations proposed to overcome such situations.

Table 1. Summary of identified adverse conditions and proposed adaptations to the CW technique [33].

	Technique Steps (According to [33])	**Adverse Conditions**	**Proposed Adaptations**
1.	Identify users and design tasks.	• It is essential to have a usability expert to apply the technique.	• The expert is replaced by an HCI student under the supervision of a mentor.
2.	Convene the cognitive walkthrough.		
3.	Walk through the action sequences of each task.	• User participation is necessary to apply the technique. • Prior preparation (e.g., conceiving the task to be performed by the user) is necessary to apply the technique.	• Users participate remotely through a forum or virtual meeting. • The task to be performed by users is not defined in advance and intentionally. Instead, it takes advantage of the tasks that are usually performed by the users.
4.	Record critical information.	• It is essential to have a usability expert to apply the technique.	• The expert is replaced by an HCI student under the supervision of a mentor.
5.	Review interface improvements.		

As part of our adaptation proposal for the CW technique, we incorporated new steps to facilitate its application in the selected OSS projects. In the following, we describe the steps of the CW according to our adaptation proposal:

- In step 1, to determine the users' profiles, we designed a questionnaire in which the users register their socio-demographic information;
- In step 2, to run the pilot test, we do the following:
  a. The expert identifies the users to be evaluated using a questionnaire called "socio-demographic profile of the user";
  b. The expert proposes a sequence of steps for each task to the users;
  c. The expert calls two known users to participate in a pilot test and perform some tasks with the OSS project;
  d. The expert records the user's criticisms (design ideas or learning problems) of the interface;
  e. The expert reviews the interface, applying what was obtained in the cognitive walkthrough to help solve the problems identified by the users.
- In step 3, we formalize the invitation to the end users to participate in a virtual meeting, the link to which will be sent to their emails;
- In step 4, we ask users to complete the document "user's socio-demographic profile", which must be completed before evaluating;
- In step 5, we ask the users to execute the tasks implemented for the evaluation. During the virtual meeting, the usability expert will manage the cognitive walkthrough and check that each task is fulfilled;
- In step 6, we designed a format for recording problems obtained during the cognitive walkthrough to the users;
- In step 7, we review the OSS project interface and propose improvements to solve the problems encountered based on the user's criticisms or opinions.

4.2. Formal Usability Test Technique and Its Adaptations

4.2.1. Description of the Formal Usability Test Technique

One of the most critical usability testing methods is FUT [34]. It corresponds to the most common tests in the HCI area, in which the user is asked to perform a series of tasks with the system prototype to observe what problems they experience, what errors they make, and how they recover from them. According to Mayhew [34], the FUT requires the participation of at least three users. It consists of 14 steps corresponding to the formal test's planning, preparation, and conduct: (i) In step 1, define the test's focus. (ii) In step 2, define the type of user and tests. (iii) In step 3, design the tasks for testing. (iv) In step 4, design materials for the test. (v) In step 5, design and set up the test environment. (vi) In step 6, recruit users for the pilot test. (vii) In step 7, execute the pilot test with the recruited users. (viii) In step 8, review the procedures and materials for the test. (ix) In step 9, recruit users for the final test. (x) In step 10, run the final test and collect data. (xi) In step 11, summarize the data recorded during the test. (xii) In step 12, analyze the data. (xiii) In step 13, draw conclusions. (xiv) In step 14, present the results.

4.2.2. Adaptations of Formal Usability Test Technique

The FUT proposed by Mayhew [34] allows evaluation of the effectiveness, efficiency, and user satisfaction with user interfaces and proposes a series of steps and tasks for its execution. However, applying the FUT directly to this type of project is impossible due to the characteristics of OSS projects. Therefore, the FUT requires adaptations since adverse conditions hinder its incorporation in OSS developments. In particular, two necessary adaptations are described at the beginning of Section 4. According to Mayhew [34], at least three users must participate to execute this technique positively. In this research work, we adapted the FUT to allow its incorporation into the selected OSS project. Table 2 summarizes the adverse conditions identified and the proposed adaptations to apply the FUT in the selected OSS project.

Table 2. Adverse conditions were identified, and proposed adaptations to the futuristic technique [34].

	Technique Steps (According to [34])	**Adverse Conditions**	**Proposed Adaptations**
1.	Define the test approach	• It is essential to have a usability expert to apply the technique.	• The expert is replaced by an HCI student under the supervision of a mentor.
2.	Define user type and tests for each type		
3.	Designing test tasks		
4.	Design test materials		
5.	Design and install the test environment		
6.	Recruit users for the pilot test	• It is essential to have a usability expert to apply the technique. • The physical presence of the users is necessary to apply the technique.	• The expert is replaced by an HCI student under the supervision of a mentor. • Users participate remotely through a forum or in virtual meetings (video calls).
7.	Execute the pilot test		
8.	Review test procedures and materials	• It is essential to have a usability expert to apply the technique.	• The expert is replaced by an HCI student under the supervision of a mentor.

Table 2. *Cont.*

	Technique Steps (According to [34])	Adverse Conditions	Proposed Adaptations
9.	Recruit end-users	• It is essential to have a usability expert to apply the technique. • The physical participation of users is necessary to apply the technique.	• The expert is replaced by an HCI student under the supervision of a mentor. • Users participate remotely through a forum or in virtual meetings (video calls).
10.	Run final test and collect data		
11.	Summarize data	• It is essential to have a usability expert to apply the technique.	• The expert is replaced by an HCI student under the supervision of a mentor.
12.	Analyze the data		
13.	Drawing conclusions		
14	Present results		

Table 3 details the steps and tasks of the FUT to be applied in an OSS project. Some steps of the original technique have been unified to facilitate its application in the OSS project. We unified steps 6 (recruit users for pilot testing) and 9 (recruit final test users) into a single step called recruit users for pilot and final testing. We describe the steps of the adapted FUT as follows:

- Define the focus of the test. Select the type of test (to measure ease of learning or ease of use);
- Define the type of user and tests. Based on the approach in the previous step, we select the types of users and general tasks for the test;
- Design test tasks. We precisely and specifically write the steps of the tasks to be executed;
- Design materials for the test. Among the materials, we find the preparation of the instructions for the participants, the welcome of the participants to the test, the introduction with the objectives of the test, and the pre-test document to define the roles of the users, the image capture permission form, the guide to the steps to be performed during the test, the error collection document, and the post-test questionnaire;
- Design and install the test environment. We prepare the tool that we will use to conduct the virtual session and the tool to record the session. We must check that these tools are available and accessible to all participating users;
- Recruit users for the pilot and final tests. We recruit one to three users for the pilot test and three to ten for the final test. Users can be recruited through the OSS project forum or by e-mail;
- Execute the pilot test. We run the pilot test and note the errors detected;
- Review test procedures and materials. If necessary, we will correct the drawbacks presented in the pilot test and improve the documents created by the research team;
- Execute the final test and collect data. First, users must perform the pilot test, for which we deliver the permission to capture images via email. Secondly, we proceed to perform the final test. For this final test, we define the following steps: (i) welcome the users, (ii) detail the introduction to the test, and (iii) collect the test data in the data collection forms. Third, we deliver the post-test questionnaires once the users finish the final usability test;
- Summarize the data. We make a report summarizing the data recorded during the test; here, we include the documents/tools used to collect the data during the test;
- Analyze the data. We interpret the data from the previous step report, considering the problems presented during the test's execution;

- Draw conclusions. We make the final report with the findings obtained after interpreting the data, including the severity of the project from the point of view of usability. We also present alternative solutions to the problems and interface design flaws;
- Present the results. We delivered the summary of the data, the analysis, the conclusions, and the evidence of the tests performed (video recordings) to the OSS project developers.

Table 3. Steps and tasks of the FUT technique to be applied in an OSS project.

	Steps	**Tasks**
1.	Define the test approach	• We selected the type of test (i) to measure the ease of learning or (ii) to measure the ease of use of the project.
2.	Define user type and tests	• We define the type of users for the test. • We list the general tasks for the test.
3.	Designing test tasks	• We describe the tasks to be performed in the test.
4.	Design test materials	• We define materials/documents to be used to perform the test. • We design the documents to be used to perform the test.
5.	Design and install the test environment	• We prepared the virtual meeting with the Google Meet tool. • We prepare the OBS tool to record the session.
6.	Recruiting users for pilot and final testing	• We recruited 1 to 3 users for the pilot test. • We recruit 3 to 10 users for the final test.
7.	Execute the pilot test	• We ran the pilot test. • We recorded the errors that occurred during the test.
8.	Review test procedures and materials	• We corrected the errors reported in the previous step.
9.	Run final test and collect data	• We run the final tests. • We collect data with the supporting material for the test.
10.	Summarize data	• We summarize the test results in a report. • We collect all the material and consolidate it.
11.	Analyze the data	• We performed the analysis of the results obtained. • It is necessary to emphasize the problems identified, understand them, and think of a solution.
12.	Drawing conclusions	• We prepare the final report on the results of the data analysis.
13.	Present the results	• We consolidate data reports, analysis, conclusions, and test evidence and present them to the project developers.

4.3. Thinking Aloud Technique and Its Adaptations

4.3.1. Description of the Technique

TA may be the most valuable tool for usability evaluation. It involves having a test subject use the software system while continuously talking out loud [51]. TA is one of the most well-known techniques, as it can help us obtain large amounts of information about a software system and its quality. To perform TA effectively, it is necessary to follow a series

of steps proposed by Nielsen [13], which are described as follows: (i) Step 1. Users must be chosen. (ii) Step 2. Initially, the activity's objective should be explained to the users. (iii) Step 3. The number of tasks must be specified. (iv) Step 4. Task execution. (v) Step 5. It is necessary to record what the evaluated user says. (vi) Step 6 consists of analyzing the results.

4.3.2. Adaptations to the Technique

When executing the steps of TA in the OSS context, specific problems may arise because it is intended for something other than this type of development. We identified two adaptations needed to incorporate AT, which are described at the beginning of Section 4. Table 4 presents the adverse conditions and adaptations needed to overcome such situations and apply TA in an OSS project.

Table 4. Summary of identified adverse conditions and proposed adaptations for the TA technique.

	Technique Steps [13]	Adverse Conditions	Proposed Adaptations
1.	Select users	• User participation is necessary to apply the technique.	• The users are friends and colleagues from the university.
2.	Explain to users	• It is necessary that several users are physically together.	• The users are friends and colleagues from the university.
3.	Determine the tasks	• It is essential to have a usability expert to apply the technique.	• The expert is replaced by an HCI student under the supervision of a mentor.
4.	Execute the task	• It is necessary that several users are physically together.	• The meeting will be virtual through one of the platforms: Google Meet or Zoom.
5.	Take notes of what the evaluated person says	• It is necessary that several users are physically together.	• The meeting will be virtual through one of the platforms: Google Meet or Zoom.
6.	Analyze the results	• It is essential to have a usability expert to apply the technique.	• The expert is replaced by an HCI student under the supervision of a mentor.

Here are the steps to apply the adapted TA to the selected OSS project:

- In step 1, we filtered user participation according to the type of test to be performed. For this project, we considered creating the survey using Google Forms; this form can be found in the link survey;
- In step 2, we created a pilot test to preliminarily run the test with users who are friends and colleagues of the university to avoid mistakes in the final test. This step is of the utmost importance, as it helps us polish the meeting that will be held with the actual participating users;
- In step 3, we conducted a video conference via Google Meet, which was recorded. At the beginning of the virtual meeting, we explained the objectives and scope of the meeting to the test participants. This virtual meeting can be conducted synchronously, allowing direct user interaction, facilitating real-time communication, and resolving any queries or concerns during the virtual conference;
- In step 4, it is necessary to specify the number of tasks to be performed and their steps. These depend on each user, considering the time they can devote to the meeting. In

our case, we have specified the use of three tasks. In addition, before the meeting, we requested the completion of a consent form to use images for the project case study;
- In step 5, users complete the tasks aloud, giving constructive criticism;
- In step 6, we perform the detailed logging of the interactions and comments of each participating user. The usability expert plays a crucial role in closely following the user contributions, capturing the most relevant aspects of using the software system under evaluation. We meticulously document the information collected in a template designed for this purpose. In addition, we developed two additional templates: one to record the time spent by each user in completing the tasks and another for users to document their own experiences and observations;
- In step 7, we analyze the results. At the end of the meeting, the expert evaluator should understand why the user acted in a specific way and not as planned. In addition, they should propose solutions that will help improve the system.

5. Design and Plan the Multiple Case Study

We investigate the incorporation of usability techniques with adaptations in OSS projects. The study design is non-experimental, as we did not randomly assign subjects or control the study groups. Because not all OSS projects have the same characteristics, minimizing the effects of external factors (e.g., geographical distribution of their members) is impossible, and evaluation by experiment is ruled out. We posed the following research question (RQ): How can CW, FUT, and TA usability techniques be incorporated into the OSS development process? Each technique was applied to a specific OSS project to explore its effectiveness. This approach allowed us to investigate their feasibility and adaptability within distinct OSS contexts.

5.1. Case Study Projects

Our research focused on projects related to e-learning, specifically those that can contribute to the support of organizations and institutions of secondary education in Ecuador. These three OSS projects allowed us to evaluate the feasibility of the proposed adaptations for the three techniques related to usability evaluation activities in different environments and contexts. The selected case studies represent tools that can support and enhance teaching and learning in virtual environments, ranging from programming to content organization and design of educational materials. The following describes the critical characteristics of the OSS projects selected to apply usability techniques.

5.1.1. Case 1. Code::Blocks

The Code::Blocks project is one of the most popular OSS projects concerning cross-platform IDEs for programming in C++ and C. It is designed to be extensible and fully configurable. In the Code::Blocks project forum, users comment on some difficulties encountered and criticize the user interface, such as (i) there being no default path when creating a project; (ii) the tool being only in English and not all users speaking English; and (iii) when there are five projects open simultaneously, and you want to run the fourth project, it is not possible, forcing you to close all other projects.

5.1.2. Case 2. Freeplane

Freeplane is a software for mind mapping and project management. In the Freeplane project forum, users comment on some difficulties encountered and criticisms of the user interface, such as (i) there being no direct relationship between an icon and what it does, (ii) the layout of some options in the menus being somewhat confusing, and (iii) the operation of some options being not very intuitive.

5.1.3. Case 3. Scribus

The graphical interface of Scribus is different from the conventional; that is to say, it is far from the standard graphical interfaces that we can observe in various projects of the same kind. For example, when we edit any property of an object within Microsoft Word, an index with the options enabled for that object is displayed at the top. In the case of Scribus, an interface called "properties" is used, where all the tools are arranged and enabled in a cascading fashion, depending on the object to be edited. In addition, the icons have no information at first glance.

5.2. Preparation and Data Collection

The data collection process is identical for CW, FUT, and TA. First, we contacted the OSS project administrators to request their collaboration in implementing the techniques. Second, to execute them, we considered that the user knew how the tool worked, and we applied similar procedures. Third, we previously evaluated the selected OSS projects to identify existing usability problems and obtain inputs to perform a new interface design. Regarding user recruitment, we disseminated the invitation to participate in applying CW, FUT, and TA through the official OSS community forums. However, in order to achieve greater user participation, we promoted its application through contact with friends and acquaintances—in our case, university students. In our research, we created web artifacts to apply the adapted usability techniques and improve communication with the members of the OSS communities. We use the official forum of the OSS projects to invite users to participate in applying the CW, FUT, and TA. We propose using a virtual meeting for remote user participation in pilot and final testing in CW, FUT, and TA. In addition, we recorded the user's involvement in templates (e.g., we reported the usability problems identified) when performing the tasks previously defined by the researchers. Also, we created an online survey to learn the profile of the participating subjects regarding the tool evaluated with the CW and TA. The web artifacts allowed us to create a virtual meeting point with OSS users to apply the techniques because these users are geographically distributed worldwide. With the creation of these web artifacts, we sought to obtain results independently for each case study because each technique requires specific information.

The data collected in the three case studies are primarily qualitative. Due to the different characteristics of the selected OSS applications and the availability of the users in these OSS projects, as most of them are volunteers and do not have free time, the data collection methods used were varied. To avoid limiting the data interpretation results, we employed various sources of information. For example, we collected data provided through e-mail, virtual meetings, and online surveys from users in each OSS project.

5.3. Data Analysis and Reports

To perform the data analysis in our multiple case study, we created tables and forms that synthesize the information obtained through online questionnaires, virtual meetings, interviews, observation, and video calls. Since the HCI does not prescribe the types of formal documents or specific tools for data collection [9], therefore, while applying the usability techniques, we proposed several tables and forms to record and present the information obtained in each case study. While the tables and forms used were adequate to record and present the information obtained in this study's limited context with few participants, we recognize that more than these tools may be required in large-scale studies with geographically distributed users. In such cases, implementing more advanced solutions, such as online platforms for data collection (e.g., Google Forms) or centralized data management systems (e.g., Microsoft Azure Data Factory), would be necessary for more efficient and scalable analysis.

We analyzed the data from each case study to answer the research question. We do not perform a statistical analysis because of the qualitative nature of the data collected in each case study. However, we provide details on the adaptations (see Section 4) to the techniques for incorporation into OSS projects. We developed each case study independently to yield specific conclusions and contribute to the study. To elaborate on the general basis of our study, it is essential to accurately describe the information obtained from users, developers, and designers of the selected OSS projects. We report the results of each of the case studies in Section 6.

6. Results of the Multiple Case Study

The case study results for the OSS projects are presented, showcasing the application of the adapted usability techniques. The implementation materials (forms, invitations, etc.) used are available at 10.6084/m9.figshare.28020965.

6.1. Results of the Code::Blocks Case by Applying CW

Regarding the Code::Bocks case study, we first sent a letter requesting authorization to participate in the OSS project to the responsible developer's email and asking for a list of potential users who could participate in this study. Due to the lack of collaboration from the OSS community to participate in applying the CW, both in the pilot and final tests, we conducted these tests with students from the Universidad Técnica Estatal de Quevedo (UTEQ). To start the pilot test, we invited two users by e-mail, attaching a link to the "User's Sociodemographic Profile" questionnaire. In this questionnaire, users had to register their data and answer questions about the Code::Blocks project.

Once the users were identified, we formally invited each of them, through their emails, to a virtual meeting to apply the CW technique. Once the pilot test was completed, with the results obtained, we identified several difficulties, such as (i) a lack of user knowledge when using the software and (ii) the test duration needing to be longer to perform the tasks. In the following, we describe the results of the pilot test. Two users participated in the pilot test and completed the "User's Sociodemographic Profile" questionnaire. The sociodemographic profile is crucial in usability evaluations to contextualize the results and understand how different characteristics, such as age or technological experience, affect the interaction with the software. In addition, it allows us to ensure that the sample of users is representative of the target population, which is vital for generalizing the results. On the one hand, the first user had problems when creating a project in the C++ programming language because the application did not request a destination path where the project would be saved. On the other hand, the second user had experience with tools similar to Code::Blocks, such as Visual Studio and NetBeans.

As a final result of the pilot test, we recorded the users' problems during the virtual meeting in the corresponding document. This document has the following information: the name of the expert, the date of the evaluation, the name of the user, the type of user, problems in the OSS, and possible solutions to the identified issues of the OSS project. It is essential to mention that only two problems were identified because the pilot test was conducted with two participants, which limited the number of possible observations during the process.

In the case of the final test, the step of identifying user profiles was omitted because we already had enough information about each of them, such as their experience (sixth-semester students of Software Engineering). Seven users participated in this final test. We made the invitation to participate via e-mail, in which we indicated the objective of the users' participation and the tasks they had to perform in a maximum time of 20 min. In addition, we attached in the same e-mail (i) the consent forms for image capture and video

recording and (ii) the document with the tasks to be performed by the users during the test. We conduct general training to start the usability evaluation, in which the expert briefly explains the cognitive walkthrough. In this training, we welcome the participants and thank them for accepting the invitation to participate in the virtual meeting. We also present an introductory summary of the CW technique and indicate what each participant must do in Code::Blocks. Due to the time constraints available to the participants to perform the evaluation, we selected two tasks: (i) create a project in C++ language and (ii) execute the script created in the project to perform the final evaluation.

According to Wharton et al. [33], the CW is carried out with expert users or users with a medium level of experience, and indeed, the final test users met this profile. During the virtual meeting, we found that users had a smooth interaction with Code::Blocks and that the time to perform the tasks ranged from 6 to 12 min. To conclude the usability evaluation, we asked the participants for their criticisms regarding the problems encountered and possible improvement proposals for the Code::Blocks project in future updates.

As a final evaluation result, we identified certain inconveniences that users expressed during the virtual meeting, which were recorded in the document "Format Instrument with tasks for users", which contains the tasks to be performed. The following are some of the inconveniences encountered by the participants during the final usability test: (i) Difficulty in identifying the execution button because there are two very similar buttons. (ii) Problems in creating a new project caused by the lack of a default path where the created projects are saved. (iii) There are too many options to do the same thing or something straightforward, like how to create a project. (iv) The current interface could be more intuitive and eye-catching, but it may not adequately capture users' attention when downloading the OSS project.

The following are some improvements proposed by users that the Code::Blocks project developer could consider:

1. Update the icons for a better visual appreciation since Code::Blocks has ancient icons;
2. Use only a few options when creating a project, as OSS projects in this category usually involve a few steps;
3. The console should be executed at the bottom of the project. Currently, a screen takes up a lot of space and makes the user's work more difficult;
4. Reduce the number of icons in the project interface, leaving only the icons of the most used functionalities, such as commenting and uncommenting code;
5. Consider a digital repository to store all the projects created to avoid the problems of assigning default paths.

Finally, we completed the document recording the problems identified during the virtual meeting with the users. In this document, we recorded the name of the expert, date of evaluation, names of the users, type of user, problems in the OSS project, and possible improvements for the OSS project.

Applying the CW technique in the Code::Blocks project revealed several significant usability issues that negatively affect the user experience. Among the main findings, we identified the absence of a default path when creating a project, which generates confusion and slows down the initial configuration process. In addition, the user interface is only available in English, which limits its accessibility for non-English-speaking users. Another critical problem detected is the inability to run a specific project when multiple projects are open simultaneously, forcing the user to close all other projects before continuing. These findings highlight the need for software interface and configuration improvements to facilitate a more intuitive and efficient user experience.

6.2. Results of the Freeplane by Applying FUT

We made a formal e-mail request to one of the Freeplane developers to perform the tests. After 22 days, we have not received a response from the Freeplane developer to our request. Subsequently, we posted this request on the official OSS project forum but received no response. However, two user members of the OSS community expressed their interest in us performing the research but did not mention wanting to participate in applying the FUT. In the first instance, we tried to perform the usability test with real users (i.e., users registered in the Freeplane forum); for this purpose, we published an announcement in the Freeplane project forum inviting users to be part of this study. After 16 days, we have not received a response from users in the project forum. Given the lack of collaboration from actual users of the Freeplane OSS community to perform the tests, we proceeded to recruit users by other means. These users were six students from the Software Engineering program at the UTEQ. These recruited users were considered viable since they have experience using OSS projects and believe they can use Freeplane as a support tool to perform tasks and exposure work in their daily work.

To execute the FUT, we emailed the formal invitations, the pre-test and post-test questionnaires, and the consent for image capture. We also defined a date and time for the test's execution. First, we conducted the pilot test with two student users from UTEQ, who had to make a mind map that collected all the aspects that should be considered to organize an ideal trip (i.e., a journey in which nothing fails). In this pilot test, we detected problems in the task approach and the focus of several questions in the pre-test and post-test forms. These problems were corrected before the final usability evaluation of Freeplane was performed. In this pilot test, we evidenced a serious problem foreseen by the research team: the technique requires constant feedback between the user and the expert. When working in a joint session, it is possible that several participants are talking and may interrupt the interaction between one of them and the evaluator. Therefore, as a countermeasure, we defined that only one user should participate in the testing session.

Users completed a pre-test survey before the evaluation to identify user profiles, which was conducted to learn about the users and their appreciation of the OSS project. The main results of the pre-test are summarized below:

- The age range of the users was between 22 and 30 years old;
- Concerning the area of work where the participants worked, all of them were in engineering;
- In total, 67% of the participants had a medium level of computer knowledge, and 33.3% had a high level of expertise. Thus, it can be concluded that they were people experienced in handling software in general;
- When the users were asked about making a mind map in the last month, 83% had needed to do so, and only 16% had not been required to do so in the previous month;
- Half of the users have yet to experience using mind mapping software;
- When asked about the frequency of users' use of OSS projects, the results indicate that 66.6% use them frequently, 16.7% use them very frequently, and another 16.7% hardly use them at all.

During the final evaluation, as users performed the formal usability test, participants forgot the importance of sharing their opinions regarding the tasks they were performing. However, the expert intervened by asking their views on the activities performed. In addition, sometimes users do not know how to perform an activity, so the expert provides clues. At the end of the formal usability test, we asked the participants about their experience developing the test and recommendations to solve the problems encountered. Below, we report some errors found by the participants during the formal usability test: (i) the icons do not relate to the functions they perform; (ii) the circle next to the node moves arbitrarily

from one side to the other; and (iii) when selecting the connect to node button, a dialog box to select the node to connect to is not displayed.

The FUT technique was crucial in identifying usability issues in the selected OSS project. This technique allowed us to directly observe users as they interacted with the software, revealing specific topics such as navigation difficulties, frequent errors, and confusing aspects of the interface. The application of FUT facilitated the collection of detailed user experience data, highlighting critical areas that required improvement. In addition, it provided valuable information on how real users perceive and use the software, which is essential for making accurate and effective adjustments to the interface and functionality of the OSS project.

6.3. Results of the Scribus Case by Applying TA

In the case of the Scribus project, to begin with, we made a formal request for authorization (via email) to the developers to perform usability tests. Due to the lack of collaboration from the OSS community to apply the TA, recruiting real users in the pilot and final tests was impossible. For this reason, we conducted the tests with external users, third-year Software Engineering students at UTEQ. It is essential to mention that this participation does not detract from the objectivity of the usability tests' application, nor does it trigger ambiguous and confusing tests because the researchers do not influence the users at the moment of performing the proposed tasks.

Before starting the final test, we conducted a pilot test with three users instructed in the activity. For each user, we monitored the time it took them to execute the test, taking note of the comments they made regarding the activity. All users agreed on the same points and stalled on the same parts of the activities. One of the main reasons the pilot users were slow to finish the test was the excess of interfaces that the OSS project has for navigating and accessing an option. The design of the interfaces and implementation of the icons are essential points mentioned in this test. This pilot test aimed to identify possible problems that may occur in the final test.

For the final test with the application of the TA technique adapted in the Scribus project, we invited eight users via email and WhatsApp. The profile of these users is as follows: they are Software Engineering students with experience in web design. In addition, we gave instructions for the development of the virtual meeting (e.g., install Scribus) and explained the document with the tasks to be performed. Of the eight invited users, seven confirmed their attendance at the virtual meeting. During the virtual meeting, we discussed the purpose of the usability evaluation through TA with the participants. In addition, we explained in detail the documents sent to the participants' e-mail, emphasizing that these documents served as a guide for developing the tasks. Each user performed three tasks. The execution of each task was recorded and monitored with a stopwatch to know the time it took them to accomplish it. The three activities were (i) inserting a header in a document, (ii) inserting a table, and (iii) inserting an image. It took users approximately 3 to 4 min to complete these tasks. It is essential to mention that these tasks are representative of Scribus' functionality. Users commented on the difficulties or problems they encountered with Scribus when performing these tasks. These difficulties are the following:

- The interface could be more intuitive for developing the tasks specified in the document sent to users to perform tasks with the Scribus tool;
- It is impossible to change the font size immediately; it is necessary to navigate between several options;
- There is a problem in the properties window when scrolling with the mouse wheel. Some values may change unintentionally due to a "phantom focus". This problem

occurs when, although the cursor is not over an active field, the system focuses on a previously selected field;
- Inserting an image directly from the web into the project is impossible;
- It is initially impossible to change an image's opacity. The process must be repeated to observe the change;
- The option to apply opacity to a table needs to be clarified because its description does not detail precisely what opacity will be given and because there are two similar options in the same properties window;
- Selecting the Edit Image option displays an error indicating that the function does not exist;
- The icons in the properties window and the top toolbar could be more intuitive;
- It needs to be clarified how to fit an image completely within its frame, filling all available space without leaving empty margins;
- When designing a table and entering the number of columns and rows, it is only possible to enter up to 9 columns or rows.

In this test, we observed that it took users longer than necessary to perform tasks, as the interface was not intuitive, and they had to access many shortcuts to perform such activities. In addition, we identified that the interface properties have problems (e.g., changing color, font, document formatting). Generally speaking, user participants commented that the Scribus interface could be more intuitive and indicated they felt uncomfortable because they lost time performing the tasks. Finally, the participants suggested some suggestions for improvement to overcome the drawbacks of Scribus. Some examples of proposals are the following: (i) Use familiar buttons for the user, similar to those used in Microsoft Word's graphical interface. (ii) Allow changing the text size in real time. (iii) Correct errors that occur when changing the opacity of an image and inserting tables. (iv) Eliminate functionalities with errors, such as the one that occurs when selecting the Edit Image option. (v) Use icons according to the objective of the action. (vi) Include the option to add tables using the right mouse click. (vii) Instead of turning off properties that do not correspond to a frame, it is preferable not to present them to avoid using unnecessary space.

The primary outcomes of using the TA technique in the Scribus case study included the identification of non-intuitive interfaces that caused user confusion and delays in task completion, usability issues that required additional steps or caused errors during task execution, and user frustrations and suggestions. These frustrations were related to complex navigation and unclear instructions. Meanwhile, suggestions included designing more familiar buttons and improving real-time feedback mechanisms. These results highlighted the effectiveness of the TA technique in uncovering usability issues and gathering valuable feedback directly from the user experience.

The adaptations made to our study's CW, FUT, and TA techniques have significantly impacted the results obtained. The adaptations allowed a more effective integration of these techniques in the specific contexts of the selected OSS projects (i.e., Code::Blocks, Freeplane, and Scribus). The CW adaptation, which included remote user participation and the replacement of usability experts by students under the supervision of mentors, facilitated the identification of usability problems and proposed viable solutions. In FUT, modifications allowed for a more realistic and accessible evaluation, using tools enabling online meetings and adapting procedures for geographically distributed users. Finally, adaptations in TA, such as conducting online sessions and involving students as experts under the supervision of mentors, allowed detailed data to be collected on users' interactions and thoughts in real time, providing deeper insight into usability issues. These adaptations improved the efficiency and effectiveness of usability evaluations. They demonstrated the

feasibility of applying these techniques in OSS projects, overcoming typical barriers such as geographical dispersion and lack of specialized usability resources.

Finally, our multiple case study identified several common usability problems in the Code::Blocks, Freeplane, and Scribus projects. These problems include non-intuitive interfaces, icons, and buttons outside their functions and a confusing menu structure. In addition, users needed help performing basic tasks due to the need for clear directions and the absence of default paths for saving projects. Specific problems were also reported, such as the lack of dialog boxes when connecting nodes and the need for multiple steps to complete simple tasks. In general, the low usability of these projects is attributed to an overloaded interface and the need for more consideration of user experience in the functionalities' design.

7. Framework for Usability Evaluation in the OSS Projects

In this section, we propose a framework designed for effectively integrating usability techniques in OSS projects. The proposal is based on our volunteer experience in the usability evaluation of different OSS projects [4,11]. We have structured the framework to be flexible and adaptable to diverse contexts. Each phase includes specific guides, facilitating its application in collaborative and remote environments. For example, in the Scribus project, we employed the technique TA, which allowed us to identify issues with unintuitive menu layouts and difficulties in understanding specific icons and propose improvements such as menu structures and more explicit icon descriptions to improve usability. While the framework was initially applied to three specific projects, its phases can be generalized, adapting to the unique characteristics of each OSS community. In the following, we describe the phases of the framework:

1. Preparation: In this phase, usability objectives are defined, remote collaborative tools are selected, and the necessary questionnaires and evaluation guides are prepared;
2. Formalization: The necessary steps of the usability techniques to be applied are formalized, providing detailed guidance on their application in OSS environments;
3. Remote Implementation: Adverse conditions that could hinder the application of the selected usability techniques are identified, and with this information, they are adapted. In the adaptation, tools that allow the participation of remote OSS users (e.g., videoconferencing tools) are considered. Evaluations are performed remotely, taking advantage of the usual tasks of OSS users;
4. Evaluation and Feedback: Qualitative and quantitative usability data are collected and analyzed, providing developers with direct feedback on identified problems and possible solutions. This direct involvement of developers in the process makes them feel integral to the project;
5. Result Analysis: The results obtained are synthesized and compared, highlighting the improvements and benefits of the proposed adaptations compared to traditional techniques;
6. Continuous Improvement: Based on the analysis's results, specific interface improvements are proposed and implemented. Periodic evaluations are performed to ensure continuous improvements in the usability of OSS projects.

Our framework addresses the unique conditions of OSS projects, such as remote participation and limited resources. While it was initially applied to three specific projects (Code::Blocks, Freeplane, and Scribus), its phases are structured to be flexible and adaptable to other contexts. For instance, during the remote implementation phase, we used tools like Google Meet and OBS Studio to overcome communication barriers, which led to the identification of difficulties in coordinating asynchronous tasks among geographically distributed participants and the proposal of more precise guidelines and schedules to streamline collaboration and improve communication efficiency. This practical approach

aims not only to solve immediate usability challenges but also to facilitate its replication in other OSS communities, pending broader future validation

Although our framework is grounded in specific experiences as OSS volunteers, we aimed to design its phases to be flexible enough for adoption in other contexts. Each phase was developed considering practical tools and procedures we tested in the evaluated projects, demonstrating their utility in identifying and addressing key usability issues. However, we acknowledge the need for further validations to ensure its generalization across OSS communities with different characteristics. This limitation allows future research to expand its validation and adapt the framework to new cases.

Table 5 summarizes the phases of the proposed framework for evaluating the usability of OSS projects.

Table 5. Framework for usability evaluation in the OSS projects.

	Phase	Description	Main Activities
1.	Preparation	Defining objectives, selecting tools, and preparing the evaluation environment.	• Identification of usability objectives. • Selection of remote collaborative tools. • Preparation of questionnaires and evaluation guides.
2.	Formalization	Formalize the steps of the techniques to be applied, providing a guide for their application.	• Formalize the steps that make up the technique. • Provide detailed guidance on the application of the technique in OSS environments.
3.	Remote Implementation	Identify the adverse conditions of the selected techniques and adapt them accordingly.	• Identify adverse conditions to adapt techniques. • Conduct remote evaluations. • Use of videoconferencing and collaborative software. • Adoption of common user tasks in OSS.
4.	Evaluation and Feedback	Collecting and analyzing usability data and providing feedback to developers.	• Qualitative and quantitative data collection. • Analysis of usability problems. • Providing feedback to developers.
5.	Result Analysis	Synthesize and compare the results obtained, highlighting the improvements and benefits of the adaptations.	• Compare the data obtained. • Analyze efficiency and effectiveness. • Identification of areas for improvement.
6.	Continuous Improvement	Implement improvements based on the results and continue to evaluate usability periodically.	• Propose specific improvements. • Implement interface changes. • Conduct periodic evaluations to ensure continuous improvements in usability.

8. Discussion of Results

In the HCI area, usability techniques, whose main objective is to obtain usable software, have been successfully applied to create usable software. However, they are used within the framework of a user-centered design. Therefore, it does not consider OSS communities' characteristics and development philosophy, making it necessary to adapt the techniques to allow their application in this type of development. These adaptations are based on the adverse conditions of each usability technique to be applied in OSS projects.

We identified two adverse conditions that the CW, FUT, and TA share. First, they require an expert for their application. Secondly, they necessarily need the physical participation of the users. The CW has a third adverse condition, which corresponds to the fact that prior preparation is necessary for its application, e.g., conceiving the task to be performed by the user during the assessment, and in the OSS community, the work performed is entirely voluntary and developed in the free time of its members. We overcome some adverse conditions by using certain web artifacts (e.g., an online survey, a forum, and a blog) known to the OSS community.

Concerning the CW technique, we propose three adaptations. First, we suggested that the usability expert be replaced by an HCI student under the supervision of a mentor. This represented a viable alternative to the presence of an expert, allowing practical application of the CW. Second, we propose that user participation be conducted remotely through online platforms, allowing us to broaden the scope of this study and facilitate collaboration, overcoming the geographic and logistical limitations associated with physical presence. Third, we propose to leverage the tasks that users typically perform with the OSS project rather than pre-defining such tasks. While applying the CW, we identified a significant challenge in communicating with the Code::Blocks community. The lead developer's lack of response over an extended period (16 days) evidenced the barriers to obtaining the necessary authorization to apply for the CW. However, the positive response from users recruited by email demonstrated significant interest in the CW and gave us a representative sample for evaluation. Once the CW was applied, we identified several problems related to the interface and user experience in the Code::Blocks project. The main issues included (i) difficulties in identifying key interface elements, such as the execute button; (ii) no default path to save created projects; and (iii) an overloaded interface that made it difficult to perform basic tasks. These findings underscore the importance of addressing the usability aspects of the Code::Blocks project.

Regarding the FUT technique, we identified two necessary adaptations to allow its adoption in OSS projects: (i) replacing the usability expert with an HCI student supervised by a mentor, and (ii) user participation is carried out remotely through, for example, virtual meetings. When applying the FUT, we encounter two drawbacks. First, when contacting the OSS project managers, we have yet to obtain a response to our request for authorization to perform the tests in the selected OSS projects. Second, there is a lack of collaboration among Freeplane community users (i.e., users registered in the forums) to participate in the evaluation. However, despite these problems, adapting the FUT has allowed us to incorporate it into the Freeplane project since it requires a small number of representative participants (minimum of three users) to obtain a reliable result [34]. In our case, we had the participation of six UTEQ student users. While applying the FUT, we collected several significant comments from users that provided detailed insight into the challenges encountered in the interface and the overall user experience. These comments covered various aspects, ranging from navigation to understanding the functions and visual appearance of the interface. User feedback highlighted three areas for improvement in the Freeplane project. First, there are difficulties in navigating and understanding the functions; for example, users face problems moving the root node or coloring nodes.

Second, users reported problems in visualizing-colored nodes, understanding the function of specific icons, and understanding the logic behind some actions, such as cloning nodes or connecting elements. Thirdly, users expressed the need for a clear and accessible properties section and the ability to drag and drop elements from a toolbar.

Regarding the TA technique, as with the FUT, we proposed two adaptations to allow its incorporation in OSS developments. First, replacing the expert with an HCI student supervised by a mentor, and second, we suggest that user participation be performed remotely through, for example, virtual meetings. While applying the TA, we identified two main challenges. First, difficulties associated with user recruitment. The lack of response from the leading developer of the Scribus project made it difficult to recruit users from the OSS community. In addition, we observed a lack of interest or availability on the part of the community to participate in this type of study, possibly due to time constraints or motivation. Second, we identified that many potential users needed to be more comfortable activating their cameras during the virtual sessions, which further limited the participation and availability of the user sample. Once the TA technique was applied, we identified several usability problems, such as (i) some icons do not represent the function they perform, (ii) some options are not found where users would intuitively look for them, and (iii) the interface looks unpleasant, among others.

The findings of our study provide a broader understanding of usability in OSS projects by identifying and addressing the specific adverse conditions that hinder the implementation of usability techniques in these environments. By adapting CW, FUT, and TA techniques for application in OSS projects, we have demonstrated that it is possible to overcome the inherent limitations of OSS communities, such as geographically dispersed membership and lack of dedicated usability resources. These adaptations not only facilitate usability evaluation in OSS but also highlight the importance of considering the unique characteristics of these projects when designing and implementing usability evaluations. Ultimately, our proposals contribute to better integration of usability into OSS development, promoting more intuitive and efficient interfaces that benefit a wide variety of users.

Our study demonstrates the feasibility of adapting usability evaluation techniques in OSS projects, improving the end-user experience. These adaptations allow OSS communities, often limited in resources and expertise, to implement usability practices without needing experts. By fostering a user-centered design culture and providing valuable feedback, developers can improve the software's functionality and usability, increasing user adoption and satisfaction. In addition, documenting and standardizing these processes provides a replicable framework for future evaluations, facilitating the integration of new contributions and continuous improvements to the software.

To address the challenges faced by OSS projects in integrating usability practices, we propose several adaptations: (i) We facilitate remote user participation through virtual meetings and online surveys due to the geographic dispersion of collaborators. (ii) We replace the lack of usability experts with advanced HCI students supervised by mentors, providing hands-on experience to students. (iii) We use regular tasks that users perform in the OSS project, minimizing additional burden and obtaining more relevant results. (iv) We formalize each step of the adapted evaluation process and develop detailed documentation to guide its implementation in future OSS projects. These adaptations improve usability and foster greater awareness and appreciation of usability within OSS communities.

During our research, we encountered several specific limitations: (i) Difficulty in recruiting active participants from OSS communities, which was mitigated by involving UTEQ students. (ii) Lack of resources and usability experts in these communities, which was addressed by replacing experts with HCI students under mentoring supervision. (iii) Geographic dispersion of members, which was solved by using virtual tools such as

Google Meet. (iv) Challenges in implementing traditional usability techniques, which were overcome by adapting CW, FUT, and TA techniques to the specific characteristics of OSS projects.

One of the main challenges in obtaining collaboration from the OSS community was the need for more response from the core developers and community users. During the evaluation of the Code::Blocks, Freeplane, and Scribus projects, we needed more interest or willingness to participate in usability testing. To overcome these challenges, we implemented several alternative methods for data collection. We used online surveys, forums, blogs, and virtual meetings to facilitate remote user participation, thus overcoming geographic and logistical constraints. In addition, we recruited external users, such as university students, who could participate under a mentor's supervision. These strategies not only broadened the scope of the study but also improved the validity and reliability of our results by including a more diverse and representative sample of users. The lack of collaboration in usability evaluation highlights the importance of implementing mechanisms that actively motivate members of OSS communities to participate in this type of activity. Strategies such as (i) gamification, (ii) offering recognition, (iii) providing users with tools that reduce the effort required to report suggestions and opinions on usability issues, (iv) offering financial incentives to users who provide feedback or participate in usability testing, and (v) incorporating usability practices into development processes could be effective in overcoming this barrier. In addition, formalizing usability procedures tailored to the characteristics of OSS projects could facilitate their acceptance and application by volunteer developers. Future research must consider these strategies to improve participation and, thus, the technical robustness of usability studies in the OSS context.

While our study applied different techniques to the three selected projects, we recognize that applying the same technique to all three projects could provide a more unified perspective on the applicability of the proposed framework. However, our decision to apply a different technique to each OSS project was due to the heterogeneous characteristics of the selected OSS projects, which led us to prioritize the exploration of the adaptability of the framework to different contexts and needs.

9. Study Limitations

The multiple case study, which is eminently qualitative, faces certain limitations in validating its contributions. We have yet to identify significant problems in construct validity or internal validity. However, external validity is limited because our study is based on only three cases. Additional case studies are essential to generalize the results to other OSS projects and validate the adaptations of CW, FUT, and TA. In the case of Code::Blocks, Freeplane, and Scribus, the main limitation of our study is that we adapted a single technique to be applied to a single OSS project. Therefore, more case studies are needed to validate the proposed adaptations using the selected usability techniques in other OSS projects. Regarding reliability, the study must include a broader range of OSS users, not just regular users registered in the community. We recommend exploring alternative methods of recruiting participants genuinely interested in contributing to this research, including social networks or other relevant forums. This more inclusive approach will help gain a deeper and more applicable understanding of the usability of OSS tools in various contexts.

10. Conclusions

The key takeaways of our study for improving usability in OSS projects are the following: Adapting usability techniques such as CW, FUT, and TA requires specific adjustments to be effective in OSS contexts, including remote user participation and replacing usability

experts with supervised students. An active community and constant updates are essential for successfully implementing usability techniques, while developer education and training, along with comprehensive documentation and support resources, facilitate the adoption of these techniques. These measures ensure that usability improvements are effective and sustainable over the long term.

The objective of our research work has been to evaluate the feasibility of applying CW, FUT, and TA in the OSS: Code::Blocks, Freeplane, and Scribus projects. Getting users to participate in the application of the techniques selflessly took work. As mentioned above, since the users are geographically distributed worldwide, they do not have enough time or resources. Without some incentive, it is not easy to get them to participate. On the other hand, we rely on the participation of external users (friends and colleagues of the UTEQ) because we did not obtain any response from the user communities for the projects or their developers. For this reason, we chose to recruit external users with intermediate to advanced knowledge. We achieved this by disseminating the material to gather information through email, WhatsApp, and forums.

To apply the three techniques, we created several templates that served as tools to collect specific information from the OSS case study projects. The participants in the usability evaluations showed great interest and were very candid in their opinions and feedback. Our adaptations to the three usability techniques CW, FUT, and TA proved effective in evaluating the OSS: Code::Blocks, Freeplane, and Scribus projects, respectively, despite our user communication and participation challenges. These findings highlight the importance of adapting evaluation methodologies to specific contexts and the need for close collaboration between researchers and the software development community to improve OSS projects' quality and user experience continuously.

The main adverse conditions of the CW, FUT, and TA that we identified as hindering their application in OSS developments are mainly twofold. First, a usability expert is needed to apply the technique. Secondly, there is a lack of face-to-face users. To solve these difficulties, we propose, on the one hand, that the expert be replaced, for example, by a student or group of HCI students under the supervision of a mentor. On the other hand, the participation of OSS users should be virtual. That is, users participate through remote meetings. Applying CW, FUT, and TA represents a significant advantage over others (such as heuristic evaluation and automated usability testing) because they interact personally with users facing problems in person or virtually.

In the case of the application of the CW, we suggest considering external users with an intermediate–advanced level of expertise, and preferably a participation of more than five users [33], since it may happen that one or more users do not show up during the application of the technique, either for personal reasons or technical problems. For the case of the FUT, we recommend involving at least five users to ensure a representative usability evaluation, ideally with varied profiles reflecting the diversity of end users. For the TA, it is advisable to integrate detailed individual sessions to capture users' intuitive reactions and thought processes during interaction with the OSS project.

Finally, we consider that the three OSS projects, Code::Blocks, Freeplane, and Scribus, have a long way to go to improve their usability. The developers of the OSS projects considered in our study use only common sense to achieve usability in their software developments, coinciding with what was stated by Andreasen et al. [2]. This results in the development of software with low-quality criteria. As mentioned above, OSS projects need the support of a team with experience in usability and who are willing to help them improve the quality of their software products.

For OSS communities to implement usability techniques effectively, the following practical recommendations are suggested: (i) provide training in usability techniques through

workshops and online courses, (ii) involve usability experts through collaborations and mentoring, (iii) provide accessible tools and resources for usability evaluation, (iv) integrate these techniques into the software development process, (v) maintain clear documentation on usability processes and results, and (v) foster a culture that values usability, promoting its benefits and recognizing efforts in this area. These measures ensure that usability becomes an integral and valued part of OSS project development, thereby improving the user experience and the success of the software.

The future of usability practices in OSS development is promising and full of potential for significant advances. As OSS continues to grow and become more integral in various industries, the emphasis on usability is expected to increase. This will be driven by increased collaboration between developers and user experience experts, adopting standardized usability frameworks, developing tools to support usability testing, and community-driven improvements. In addition, integrating artificial intelligence into usability testing will provide intelligent insights and predictions of user behavior. At the same time, inclusive design will ensure that OSS projects are accessible to many users. By embracing these trends and developments, the OSS community can significantly improve the usability of their projects, making them more user-friendly and accessible to a broader audience, which will contribute to the success and adoption of OSS in various sectors.

Because of the findings of this study, we have to identify four directions for future research in the area of usability improvement in OSS projects. First, it is recommended that the number of case studies be expanded to include a wider variety of OSS projects covering different domains and software types. This will allow for better validation and generalization of the results obtained. Second, it is essential to investigate the application of usability evaluation techniques beyond the three techniques studied (CW, FUT, and TA) to explore the effectiveness of other methods in the OSS context. This will allow us to broaden the spectrum of available tools and gain a more complete understanding of the user experience in these environments. Third, we also consider it vital to study further the factors that affect the OSS community's participation in usability studies. Developing effective strategies to foster collaboration and user feedback will be fundamental to improving the quality of OSS projects and their adaptation to the needs of end users. Finally, another area of interest is the development of specific tools or adaptations of existing tools to facilitate usability evaluation in OSS environments (for example, a browser extension for collecting end-user feedback directly from the OSS project interface). These tools should be designed to consider the particularities and limitations of the community and OSS projects to optimize the evaluation process and improve the user experience.

Author Contributions: Conceptualization, L.L. and J.W.C.; methodology, L.L., J.W.C., R.L. and N.R; validation L.L., J.W.C., R.L. and N.R.; supervision, J.W.C.; project administration, L.L.; funding acquisition, L.L. All authors have contributed to the writing and editing of the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the FOCICYT 2022-2023 project of the UTEQ and the DIUDA 22436 project of the Universidad de Atacama.

Institutional Review Board Statement: This work involved human subjects in its research. All ethical and experimental procedures and protocols were approved by the Research and Ethics Committee of the Universidad Técnica Estatal de Quevedo (FOCICYT 2022–2023 project) on 2 May 2023.

Informed Consent Statement: Informed consent was obtained from all subjects involved in this study.

Data Availability Statement: Data available in a publicly accessible repository (10.6084/m9.figshare.28020965).

Acknowledgments: Work supported by the FOCICYT project funded this research, "Evaluation of the usability of open-source tools for e-learning as support for the teaching process aimed at high school students in Ecuador", corresponding to the Ninth Call of the Competitive Fund of Research, Science, and Technology FOCICYT 2022-2023 project of the UTEQ, and the DIUDA 22436 project of the Universidad de Atacama.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bretthauer, D. Open source software: A history. *Inf. Technol. Libr.* **2002**, *21*, 3–10.
2. Andreasen, M.S.; Nielsen, H.V.; Schrøder, S.O.; Stage, J. Usability in open source software development: Opinions and practice. *Inf. Technol. Control* **2006**, *35*, 303–312. [CrossRef]
3. Raza, A.; Capretz, L.F.; Ahmed, F. Maintenance support in open source software projects. In Proceedings of the 8th International Conference on Digital Information Management (ICDIM'13), Islamabad, Pakistan, 10–12 September 2013; pp. 391–395. [CrossRef]
4. Llerena, L.; Rodriguez, N.; Castro, J.W.; Acuña, S.T. Adapting usability techniques for application in open source Software: A multiple case study. *Inf. Softw. Technol.* **2019**, *107*, 48–64. [CrossRef]
5. Sánchez, W. La usabilidad en ingeniería de software: Definición y características. *Innovación. Rep. De Investig.* **2011**, *2*, 7–21.
6. Almazroi, A.A. A systematic mapping study of software usability studies. *Int. J. Adv. Comput. Sci. Appl.* **2021**, *12*, 228–241. [CrossRef]
7. Castro, J.W.; Garnica, I.; Rojas, L.A. Automated tools for usability evaluation: A systematic mapping study. In *Social Computing and Social Media: Design, User Experience and Impact*; Meiselwitz, G., Ed.; HCII 2022; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2022; Volume 13315, pp. 28–46. [CrossRef]
8. Ren, R.; Castro, J.W.; Santos, A.; Dieste, O.; Acuna, S.T. Using the SOCIO chatbot for UML modelling: A family of experiments. *IEEE Trans. Softw. Eng.* **2023**, *49*, 364–383. [CrossRef]
9. Ferré, X.; Moreno, A.M. Integración de la IPO en el proceso de desarrollo de la ingeniería del software: Propuestas existentes y temas a resolver. In Proceedings of the V Congreso Interacción Persona-Ordenador (Interacción'04), Lleida, Spain, 3–7 May 2004; pp. 134–141.
10. Çetin, G.; Göktürk, M. Assessing usability readiness of collaborative projects. *Comput. Syst. Sci. Eng.* **2011**, *26*, 259–274.
11. Llerena, L.; Castro, J.W.; Acuña, S.T. A pilot empirical study of applying a usability technique in an open source software project. *Inf. Softw. Technol.* **2019**, *106*, 122–125. [CrossRef]
12. Castro, J.W. Incorporación de La Usabilidad En El Proceso de Desarrollo Opencas Source Software. Doctoral Thesis, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, España, 2014.
13. Nielsen, J. *Usability Engineering*; Morgan Kaufmann: Burlington, MA, USA, 1993; ISBN 0-12-518405-0.
14. Messerschmitt, D.G. Back to the user [open source]. *IEEE Softw.* **2004**, *21*, 89–91. [CrossRef]
15. Ćetin, G.; Mehmet, G. A measurement based framework for assessment of usability-centricness of open source software projects. In Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems (SITIS'08), Bali, Indonesia, 30 November–3 December 2008; pp. 585–592. [CrossRef]
16. Terry, M.; Kay, M.; Lafreniere, B. Perceptions and practices of usability in the free/open source software (FOSS) community. In Proceedings of the 28th International Conference on Human Factors in Computing Systems (CHI'10), Atlanta, GA, USA, 10–15 April 2010; pp. 999–1008. [CrossRef]
17. Rajanen, M.; Iivari, N. Examining usability work and culture in OSS. In *Open Source Systems: Adoption and Impact*; Damiani, E., Frati, F., Riehle, D., Wasserman, A., Eds.; OSS 2015; IFIP Advances in Information and Communication Technology; Springer: Cham, Switzerland, 2015; Volume 451, pp. 58–67. [CrossRef]
18. Çetin, G.; Verzulli, D.; Frings, S. An analysis of involvement of HCI experts in distributed software development: Practical issues. In *Online Communities on Social Computing*; Shuler, D., Ed.; OCSC 2007; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4564, pp. 32–40. [CrossRef]
19. Hedberg, H.; Iivari, N.; Rajanen, M.; Harjumaa, L. Assuring quality and usability in open source software development. In Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops'07), Minneapolis, MN, USA, 20–26 May 2007; pp. 1–5. [CrossRef]
20. Nichols, D.M.; Twidale, M.B. The usability of open source software. *First Monday* **2003**, *8*, 1. [CrossRef]
21. Ternauciuc, A.; Vasiu, R. Testing usability in Moodle: When and how to do it. In Proceedings of the 2015 IEEE 13th International Symposium on Intelligent Systems and Informatics (SISY'15), Subotica, Serbia, 17–19 September 2015; pp. 263–268. [CrossRef]
22. Assal, H.; Chiasson, S.; Biddle, R. Cesar: Visual representation of source code vulnerabilities. In Proceedings of the 2016 IEEE Symposium on Visualization for Cyber Security (VizSec'16), Baltimore, MD, USA, 24 November 2016; pp. 1–8. [CrossRef]

23. Ledesma, A.; Al-Musawi, M.; Nieminen, H. Health figures: An open source JavaScript library for health data visualization. *BMC Med. Inform. Decis. Mak.* **2016**, *16*, 38. [CrossRef] [PubMed]
24. Khalajzadeh, H.; Simmons, A.J.; Abdelrazek, M.; Grundy, J.; Hosking, J.; He, Q. An end-to-end model-based approach to support big data analytics development. *J. Comput.* **2020**, *58*, 100964. [CrossRef]
25. Weninger, M.; Grünbacher, P.; Gander, E.; Schörgenhumer, A. Evaluating an interactive memory analysis tool: Findings from a cognitive walkthrough and a user study. *ACM Human-Comput. Interact.* **2020**, *4*, 75. [CrossRef]
26. Tegos, S.; Mavridis, A.; Demetriadis, S. PeerTalk: Enabling agent-enhanced real-time collaboration in MOOCs. In Proceedings of the ACM 24th Pan-Hellenic Conference on Informatics (PCI'20), Athens, Greece, 20–22 November 2020; pp. 152–155. [CrossRef]
27. Cemellini, B.; Thompson, R.; Van Oosterom, P.; Vries, M. Usability testing of a web-based 3D Cadastral visualization system. In Proceedings of the 6th International FIG Workshop on 3D Cadastres, Delft, The Netherlands, 2–4 October 2018; pp. 529–548.
28. Hall, J. Usability Themes in Open Source Software. Ph.D. Thesis, University of Minnesota, Minneapolis, MN, USA, 2014.
29. Pietrobon, R.; Shah, A.; Kuo, P.; Harker, M.; McCready, M.; Butler, C.; Martins, H.; Moorman, C.T.; Jacobs, D.O. Duke surgery research central: An open-source web application for the improvement of compliance with research regulation. *BMC Med. Inform. Decis. Mak.* **2006**, *6*, 32. [CrossRef] [PubMed]
30. Gujrati, S.; Vasserman, E. The usability of truecrypt, or how I learned to stop whining and fix an interface. In Proceedings of the Third ACM Conference on Data and Application Security and Privacy (CODASPY'13), San Antonio, TX, USA, 18–20 February 2013; pp. 83–84. [CrossRef]
31. Brun, D.; Ferreira, S.M.; Gouin-Vallerand, C.; George, S. CARTON project: Do-it-yourself approach to turn a smartphone into a smart eyewear. In Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media (MoMM'16), Singapore, Singapore, 28–30 November 2016; pp. 128–136. [CrossRef]
32. Gordillo, A.; Barra, E.; Quemada, J. An easy to use open source authoring tool to create effective and reusable learning objects. *Comput. Appl. Eng. Educ.* **2017**, *25*, 188–199. [CrossRef]
33. Wharton, C.; Rieman, J.; Lewis, C.; Polson, P. *The Cognitive Walkthrough Method: A Practitioner's Guide*; John Wiley & Sons, Inc.: New York, NY, USA, 1994.
34. Mayhew, D. *The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design*, 1st ed.; Morgan Kaufmann: Burlington, MA, USA, 1999.
35. SourceForge. Available online: https://sourceforge.net/ (accessed on 15 September 2023).
36. Nichols, D.M.; Twidale, M.B. Usability processes in open source projects. *Softw. Process Improv. Pract.* **2006**, *11*, 149–162. [CrossRef]
37. Bitzer, J.; Schrettl, W.; Schröder, P.J.H. Intrinsic motivation in open source software development. *J. Comp. Econ.* **2007**, *35*, 160–169. [CrossRef]
38. Benson, C.; Müller-Prove, M.; Mzourek, J. Professional usability in open source projects: GNOME, OpenOffice.org, NetBeans. In Proceedings of the Extended Abstracts on Human Factors in Computing Systems (CHI EA'04: CHI'04), Vienna, Austria, 24–29 April 2004; pp. 1083–1084. [CrossRef]
39. Rajanen, M.; Iivari, N. Open source and human computer interaction philosophies in open source projects—Incompatible or co- existent? In Proceedings of the International Conference on Making Sense of Converging Media (AcademicMindTrek'13), Tampere, Finland, 1–4 October 2013; pp. 67–74. [CrossRef]
40. Northrop, E.E.; Lipford, H.R. Exploring the usability of open source network forensic tools. In Proceedings of the 2014 ACM Workshop on Security Information Workers (SIW'14), Scottsdale, AZ, USA, 7 November 2014; pp. 1–8. [CrossRef]
41. Hall, J. The usability of GNOME. *Linux J.* **2014**, *2014*, 3.
42. Rajanen, M.; Iivari, N. Power, empowerment and open source usability. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15), Seoul, Republic of Korea, 18–23 April 2015; pp. 3413–3422. [CrossRef]
43. Nielsen, L.; Bødker, M. To do or not to do: Usability in open source development. *Interfaces* **2007**, *71*, 10–11.
44. Runeson, P.; Höst, M.; Rainer, A.; Regnell, B. *Case Study Research in Software Engineering: Guidelines and Examples*; John Wiley & Sons: Hoboken, NJ, USA, 2012; ISBN 9781118104354.
45. Runeson, P.; Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empir. Softw. Eng.* **2009**, *14*, 131–164. [CrossRef]
46. Genero Bocco, M.F.; José Antonio, C.L.; Piattini Velthuis, M.G. *Métodos de Investigación En Ingeniería Del Software*; Grupo Editorial Ra-Ma: Madrid, Spain, 2014; ISBN 978-84-9964-507-0.
47. Yin, R.K. *Case Study Research: Design and Methods*, 5th ed.; SAGE Publications, Inc.: Thousand Oaks, CA, USA, 2013.
48. Sorbello, A.; Ripple, A.; Tonning, J.; Munoz, M.; Hasan, R.; Ly, T.; Francis, H.; Bodenreider, O. Harnessing scientific literature reports for pharmacovigilance: Prototype software analytical tool development and usability testing. *Appl. Clin. Inform.* **2017**, *8*, 291–305. [CrossRef] [PubMed]
49. Constantine, L.L.; Lockwood, L.A.D. *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*, 1st ed.; Addison-Wesley: Boston, MA, USA, 1999.

50. Dubé, L.; Paré, G. Rigor in information systems positivist case research: Current practices, trends, and recommendations. *MIS Q.* **2003**, *27*, 597–635. [CrossRef]
51. Lewis, C. *Using the "Thinking-Aloud" Method in Cognitive Interface Design*; IBM T.J. Watson Research Division: New York, NY, USA, 1982.