BOCADE: Improving the User Experience (UX) of Competitive Programming Contestants with a Visual Studio Code Extension

Luiz Gustavo Albuquerque dos Santos[1], **Fabiola G. C. Ribeiro**[1], **Kenia S. de Oliveira**[2]

[1]Department of Information Systems
Federal Institute Goiano – Catalão, GO – Brazil

[2]Federal Institute of Brasília – Brasília, DF – Brazil

gusalbukrk@outlook.com,
fabiola.ribeiro@ifgoiano.edu.br, keniasoli@gmail.com

Abstract. *The BOCA programming contest management system provides a web interface for contestants. Moreover, contestants use standard code editors to develop their code. However, these editors are not optimized for the specific context of use of competitive programming. Therefore, in this paper, we present BOCADE, an extension for Visual Studio Code with the following functionalities: built-in BOCA interface and PDF viewer, automated test cases extraction and execution panel, and a mechanism for quick workspace organization. To evaluate the extension, two usability tests were conducted — a formative and a summative. The results suggests that the extension provides a superior UX for competitive programming contestants compared to the traditional setup.*

1. Introduction

It is well-established in the academic literature that computer programming is a cognitively complex activity and presents a significant challenge to students to learn and teachers to instruct effectively [Xia 2017, Flórez et al. 2017]. Traditional pedagogical approaches, such as theoretical exposition and emphasis on programming language syntax, are prevalent but show limited efficacy in building practical coding skills and deep understanding [Cheah 2020]. On the other hand, a practical approach with hands-on exercises like coding challenges is known to improve student motivation and performance [Xia 2017]. Another commonly utilized teaching strategy is gamification. [Tenório and Bittencourt 2016] defines gamification as a pedagogical methodology that leverages game-like elements (e.g., context, fast feedback, competition, achievements, etc.) to promote learning and facilitate creative problem-solving. The potential of gamification to improve computer programming learning is supported by empirical research [Zhan et al. 2022, Rodrigues and Isotani 2023, do Carmo Nogueira et al. 2018].

Competitive programming is a pedagogical activity that blends hands-on problem-solving with gamified elements, such as rapid feedback, leaderboards, and problems presented with engaging narratives [Moreno and Pineda 2018, Coore and Fokum 2019]. Another well-regarded programming teaching strategy often present in programming contests is collaborative coding [Sentance and Csizmadia 2017, Pham and Nguyen 2019]. Empirical evidence suggests a positive correlation between participation in programming contests and desirable student outcomes, including higher engagement and

DOI: 10.5753/sbie.2024.241858

enhanced computational thinking and programming skills [Moreno and Pineda 2018, Piekarski et al. 2015, Silva et al. 2023, Yuen et al. 2023, Audrito et al. 2023].

Many competitive programming events occur around the world and the exact format of the contests varies according to the organizer. However, at its core, competitive programming involves writing computer programs to solve a set of problems, while simultaneously competing with other programmers based on parameters such as program correctness, execution time, and development time [Majumdar 2017]. The International Collegiate Programming Contest (ICPC) is one of the most renowned programming competitions in the world [ICPC Foundation 2024]. In Brazil, the Brazilian Computing Society's (SBC) Programming Marathon is designed for undergraduate and early graduate students. It comprises two phases: regionals and finals, with the finals also serving as part of the Latin American regional qualifier for the annual ICPC world finals [Morais and Ribas 2019]. In addition to the official phases, smaller, non-official contests are held to help prepare participants for the Programming Marathon official contests [Algar Telecom 2012], or simply to incorporate a practical dimension into programming curricula in secondary and higher education [Piekarski et al. 2015, Silva et al. 2023].

In Programming Marathon-related competitions, the BOCA Online Administrator System (BOCA) serves as the contest management system, offering a web-based contestant interface. Contestants rely on the web browser not only for interacting with the BOCA interface but also for viewing Portable Document Format (PDF) files. In addition, contestants use standard code editors to develop their code; albeit these tools are sufficient, they are not optimized for competitive programming as they lack specialized features for this context of use. Moreover, frequent switching between applications is detrimental to productivity [Murty et al. 2022]. Therefore, in this paper, we present BOCA Development Environment (BOCADE), an open-source Visual Studio Code (VS Code) [Microsoft 2024b] extension with the following functionalities: built-in BOCA interface and PDF viewer, automated test cases extraction and execution panel, and a mechanism for quick workspace organization. To evaluate the extension and assess the validity of our hypothesis regarding its potential to provide a superior UX compared to the traditional setup, we conducted two usability tests — a formative test and a summative test. Although the evaluation results present certain limitations, the findings demonstrate enhanced usability and aesthetics, along with a clear user preference for the extension over the traditional setup. As such, the results provide reasonable evidence to conclude that the extension improve the UX of competitive programming contestants.

The remainder of this text is organized as follows. Section 2 presents an examination of existing work on the development of tools within the domain of competitive programming. Section 3 outlines the key topics necessary for thorough comprehension of this paper. Section 4 provides a overview of the BOCADE extension. Section 5 focuses on the experimental methodologies, produced results and ensuing discussions about the findings. Finally, Section 6 has the concluding remarks and future work.

2. Related Works

There is a wide variety of software related to competitive programming. In this section, we exclusively concentrate on previous work in academic literature which has explored the creation or customization of software used in ICPC-styled competitions.

According to [Maggiolo and Mascellani 2012], the main technical challenges of organizing a programming contest can be categorized into three parts: (1) problem creation, including all its related metadata such as statements, solutions and test cases; (2) contestant environment configuration, in particular with respect to environment consistency and network restrictions; and (3) contest management, i.e., problem distribution, automated grading with feedback, and real-time ranking updates.

In [Maggiolo and Mascellani 2012], the authors presented the Contest Management System (CMS). It was developed to be used in the International Olympiad in Informatics (IOI) — a programming competition similar to the ICPC but directed towards secondary school students. The Programming Contest Control (PC2) is another system for contest management and it has been used in some ICPC contests [Ganorkar 2017]. Many efforts have been documented on the development of complementary tools to improve this system. Notably, [Ganorkar 2017] documented the development of the Web Team Interface client as an alternative interface to the desktop application and [Adithya 2011] developed a security sandbox to enhance code execution safety. The contest management system that has been increasingly used in recent ICPC contests — including in the past few world finals — is DOMjudge [DOMjudge 2024]. In [Pham and Nguyen 2019], the authors augmented DOMjudge's capabilities by adding a plagiarism detection system. In the SBC's Programming Marathon, BOCA [de Campos and Ferreira 2004] is used as the contest management system and Maratona Linux [Morais and Ribas 2019] for contestant environment configuration. Due to their significance for this paper, both will be examined in greater detail in the literature review.

The current section underscores the fact that the domain related to software used for ICPC-styled contests has received many contributions. Nonetheless, our research identified no prior work proposing a specialized code editor or a modification to an existing code editor aimed at optimizing the development environment for these competitions. In this paper, we propose the BOCADE to address this gap.

3. Literature Review

The present section establishes a foundation in the two key topics necessary for a thorough understanding of this paper — the BOCA contest management system and UX.

3.1. BOCA

Programming competitions can be broadly categorized by location: on-site and online. On-site competitions guarantee competition integrity through in-person proctoring and controlled environment — access to the internet or personal electronic devices is strictly prohibited. In contrast, online contests are well-suited for two contexts: programming course assignments and training for on-site competitions [Combéfis and Wautelet 2014]. ICPC-styled competitions are typically administered on-site [Pham and Nguyen 2019].

Accurate and efficient programming contest management demands a system that automates tasks like problem distribution, automated grading with feedback, and real-time ranking updates [Maggiolo and Mascellani 2012]. The academic literature lacks a single, universally agreed-upon term for these systems, though they are often referred to as contest management systems [Maggiolo and Mascellani 2012, Kostadinov et al. 2010], contest control systems [Ganorkar 2017], or online judge systems [Pham and Nguyen 2019]. Throughout this paper, we refer to these systems as contest management systems.

Among the many programming contest management systems available, BOCA stands out as a very popular alternative in Brazil, in large part due to its adoption in the ICPC-styled Programming Marathon [de Campos and Ferreira 2004]. BOCA is frequently used in conjunction with Maratona Linux as a complementary software solution for the technical needs of programming competitions. Maratona Linux is an Ubuntu-based Linux distribution that provides a comprehensive development environment for competitive programming contestants. This includes popular code editors (e.g., Emacs, Vim, gedit, Geany, CodeBlocks, VS Code, PyCharm, IntelliJ IDEA, Eclipse, CLion), along with compilers and interpreters for the languages commonly used in programming contests (i.e., C, C++, Java, Kotlin, and Python). Equally as important, Maratona Linux has the necessary safeguards to prevent any non-authorized access to the internet or physical media [Morais and Ribas 2019].

As it is usual for contest management systems, BOCA provides a contestant interface which is accessible exclusively through a web browser. It has a markedly dated visual design and its functionalities are organized into seven tabs. A brief description of the functionalities provided by each tab as described in [dos Santos 2024] is as follows: (1) problems: information about the contests problems; (2) runs: information about previous submitted solutions and a form for the submission of new solutions; (3) score: real-time contest ranking; (4) clarifications: information about submitted questions and a form for the submission of new questions; (5) tasks: forms for sending files for printing and for asking for urgent help; (6) backup: form for uploading files for backup; and (7) options: form for editing contestant information, such as username, full name, and password.

3.2. UX

In Human-Computer Interaction (HCI), usability and UX are two interrelated concepts. As defined by [International Organization for Standardization (ISO) 2018], usability is the degree to which users can accomplish specific goals with effectiveness, efficiency, and satisfaction; UX, on the other hand, encompasses how users perceive and respond to a system in a specified context of use. The overlap between the two concepts is evident. Nonetheless, UX is generally understood to be a broader concept that encompasses usability and all aspects of the interaction such as aesthetics, accessibility, and overall satisfaction [Lima et al. 2022].

Usability evaluation relies on established principles, often grouped into a set of guidelines by HCI authors (e.g., Usability Heuristics, Golden Rules of Usability, Ergonomic Criteria, etc.). [Lima and Benitti 2021] compared various guidelines sets and identified six key usability principles: (1) proper error handling; (2) consistent design; (3) user feedback for significant actions and events; (4) adaptability to user needs; (5) reduced cognitive load; and (6) maximized user control over the interface.

[Riihiaho 2018] provides a comprehensive overview of usability testing — a widely employed evaluation technique wherein representative users execute a specified set of tasks to evaluate the usability of a system under the supervision of a facilitator. Usability tests can be categorized into two primary types: formative and summative. Formative evaluation involves gathering user feedback throughout development to guide iterative improvements, while summative evaluation is conducted at the end of development

to assess whether the final product meets the usability requirements. Formative evaluations are typically shorter and less comprehensive. The recommended number of users for a usability test varies by author, but the relevant literature suggests that 5–9 users are generally sufficient to identify around 80% of the usability issues. In regards to data collection, questionnaires are frequently utilized to collect data on users' backgrounds and their experiences with the evaluated system. The use of standardized questionnaires is recommended because they offer greater reliability. The System Usability Scale (SUS) has emerged as the de facto standard questionnaire for usability evaluation. It comprises ten statements rated on a five-point Likert scale. The results are used to calculate the SUS score, a standardized metric ranging from 0 to 100, which serves as a concise indicator of a system's overall usability

According to [Lewis 2018], the Sauro-Lewis curved grading scale classifies SUS scores by assigning letter grades based on standardized score ranges, with the highest grade (A+) spanning 84.1–100 and the lowest grade (F) ranging from 0–51.6. The data employed to develop this scale yielded an average score of 68. Hence, this value has been widely adopted as the average SUS score in HCI literature. More recent studies indicate an slightly upward shift in the mean SUS score, now estimated to be around 70.8.

4. BOCADE

The primary objective of BOCADE is to improve upon the traditional setup for competitive programming. This traditional approach consists in the use of two separate tools: a web browser for interacting with the BOCA contestant interface and viewing PDF files, and a standard code editor lacking specialized functionalities for development in programming contests. The extension aims to streamline this workflow by integrating functionalities for competitive programming within a code editor. During the requirements elicitation phase, we drew upon our experience in participating in and organizing programming contests to design functionalities aimed at enhancing the UX for participants in programming competitions. A key goal is the elimination of the need for a web browser.

VS Code emerged as the ideal target platform for the extension because it is free, cross-platform, extensible, and very popular [Verma 2020, Stack Overflow 2023]. The VS Code extension Application Programming Interface (API) enables developers to customize nearly every aspect of the editor and extend its functionalities to encompass custom tools [bin Uzayr 2022]. Moreover, VS Code is included by default in Maratona Linux and it offers comprehensive support for all of the languages commonly used in programming contests, unlike many of the other development environments available in Maratona Linux that cater exclusively to specific programming languages.

4.1. Requirements

According to [Kurtanović and Maalej 2017], software requirements can be categorized into two distinct types: functional and non-functional. Functional requirements (FR) define the specific capabilities the system must offer and non-functional requirements (NFR) describe its properties and constraints (e.g., performance, security, and usability).

The first functional requirement (FR1) is the integration of the BOCA contestant interface into VS Code. The functionalities to be replicated in the extension are the ones present in the problems, runs, score, and clarifications tabs. This decision was

made because the functionalities contained in the other tabs are much less frequently used. The proposal for the FR1 stemmed from the observation that contestants in programming competitions were restricted to interacting with BOCA solely through a web browser, despite spending most of their time within a code editor. In fact, each one of the contest management system discussed in the related works section — CMS, PC2, and DOMjudge — provides access to the participant interface exclusively through a web interface, or, in the case of PC2, also via a desktop application. Nevertheless, as discussed in [Murty et al. 2022], the necessity of frequently switching between applications presents a drawback as it can negatively impact productivity due to the time required for users to adjust to the application, its semantic context, and its purpose. Research in psychology and neuroscience demonstrates that task switching is cognitively taxing. Even toggling between just two applications constitutes task switching. The frequent need to alternate between applications is particularly detrimental for programming contests participants given the time-constrained nature of competitive programming. The second functional requirement (FR2) is the implementation of an integrated PDF viewer. Such functionality is of utmost importance due to the prevalent practice of storing problem descriptions in PDF files. These first two functional requirements are related because their combined functionality eliminates the need for users to switch between applications.

A test cases panel is the third functional requirement (FR3) and its intent is to significantly expedite the assessment of code correctness. As is done currently, contest participants need to type each sample input individually and subsequently perform a mental comparison between the sample output and the output generated by their code. The test cases panel functionality aims to automate such process and it is inspired by a similar feature available in a popular VS Code extension [Agrawal 2020a] used for non-ICPC-styled online competitions. The panel should be comprised of the following elements: an interface to manage (i.e. view, create, edit, and delete) test cases, a button to extract test cases from a PDF file, and a button to run a source code file against a set of test cases.

The workspace organization button is the fourth functional requirement (4FR) and its aim is to automate the process of arranging the code editor tabs and other relevant layout elements in a way that optimizes the workspace for quick navigation in programming contests. When the workspace organization button is clicked, the test cases panel must open and the editor area must be splitted into two editor groups. The source code file tabs must be placed in the left group, and the BOCA contestant interface tab and the PDF file tabs must be placed in the right group. The inclusion of such functionality is particularly pertinent because the integration of the PDF viewer will lead users to keep multiple tabs open at once.

Lastly, the fifth functional requirement (FR5) consists in the implementation of a light/dark mode toggle button meant to facilitate user selection of their preferred mode.

As discussed in the literature review, usability and aesthetics are critical factors in UX. Thus, adherence to the key usability principles identified in the literature review constitutes the project's first non-functional requirement (NFR1) and a visually appealing interface constitutes the second non-functional requirement (NFR2). As part of the NFR2, the interface must also retain the same fundamental structure as the BOCA web contestant interface (to ensure familiarity for existing users) and provide support for both light and dark modes in the two custom layout elements contributed by the extension — i.e., the

BOCA contestant interface and the test cases panels.

4.2. Implementation

Herein, we detail the solution stack utilized to satisfy the project's requirements. It should be noted that the versions of the targeted software used during the extension development were: BOCA 1.5.17, Maratona Linux 20231006, and VS Code 1.81.1.

TypeScript [Microsoft 2024a] was chosen as the project's programming language due to its popularity and static typing capabilities. Moreover, the following libraries were used to develop the functional requirements: React [Meta Open Source 2024], Webview UI Toolkit [Microsoft 2024c], jsdom [jsdom 2024], vscode-pdf [Tomoki 2023], pdfplumber[Singer-Vine 2024] and compile-run [Agrawal 2020b].

The interface of the two custom elements contributed by the extension — the BOCA contestant interface (FR1) and the test cases panel (FR3) — were built using React and Webview UI Toolkit. React was chosen because of its widespread popularity in the domain of interface development. The component library Webview UI Toolkit was chosen for its ability to ensure cohesion both between components and with the editor's overall aesthetic, while also offering out-of-the-box support for light and dark modes (NFR2).

In regards specifically to the integrated BOCA contestant interface (FR1), an early technical challenge during its development arose from the lack of an API in BOCA, which was overcome by the use of a web scraper. As stated in [Glez-Peña et al. 2014], web data scraping is a technique commonly used when a standardized Representational State Transfer (REST) API is unavailable. In such cases, a software agent is used to mimic human web browsing behavior, systematically extracting the desired data from websites. Initially, we considered using direct database access because it would offer a slightly faster data retrieval compared to scraping. However, this approach presented a critical user perception issue. While the VS Code Extension API offers mechanisms for secure password storage, requiring contest organizers to provide their database credentials directly within the extension could be misconstrued as insecure. Therefore, the scraping approach was preferred over direct database access. Due to its popularity and straightforward API, jsdom was chosen as the solution for web scraping. In relation to the other functionality meant to eliminate the need for switching between applications (FR2), no custom code was necessary to enable the displaying of PDF files inside the code editor. A search in the VS Code marketplace revealed that the vscode-pdf extension already offered an integrated PDF viewer. Therefore, this third-party extension was incorporated to the project as a dependency, ensuring its automatic installation upon the BOCADE installation.

For the test cases panel (FR3), two libraries were employed to satisfy its requirements not related to interface development — pdfplumber for the extraction of test cases from PDF files and compile-run for the execution of the test cases against a source code file.

As for the workspace organization button (FR4) and the light/dark mode toggle button (FR5), their implementation relied solely on the VS Code extension API without the need for additional libraries.

NFR1's six key usability principles served as guiding principles throughout the development process. These principles were incorporated into the extension design as

follows: (1) proper error handling is ensured with meaningful error messages displayed in the event of any failure; (2) consistent interface elements are achieved by utilizing components from the Webview UI Toolkit; (3) user feedback is given when significant actions occur, such as success messages upon successful form submissions; (4) adaptability to user needs is bolstered by the use of the Webview UI Toolkit which provides built-in support for light and dark mode; (5) the extension reduces cognitive load by eliminating the need to keep multiple applications open and significantly streamlining the process of code correctness assessment; and, lastly, (6) the extension promotes user control by providing a diverse set of modular features.

The BOCADE interface is shown in the Figure 1. Additionally, screencasts are available on the project's repository to further showcase the extension.

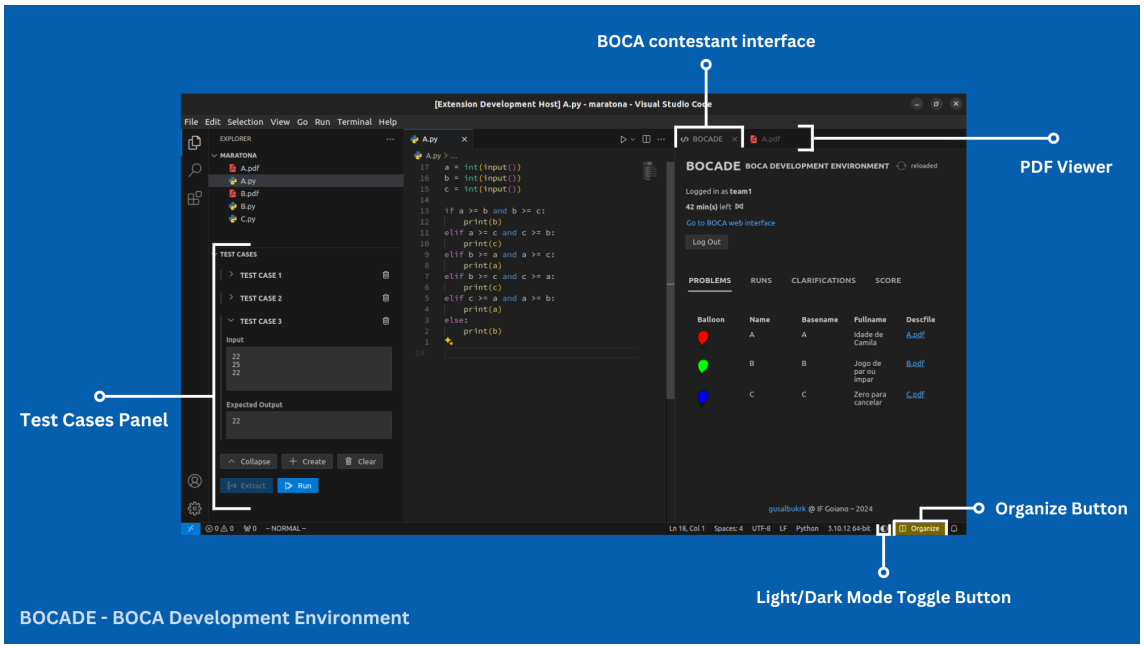

Figure 1. The BOCADE interface

5. Evaluations

This section focuses on presenting the experimental methodologies, results and discussion for each of the two usability tests employed to evaluate the BOCADE.

5.1. Formative Evaluation

The evaluation described in the present subsection was conducted not only to identify usability issues but also to validate our hypothesis concerning the potential UX benefits of the extension. It took place at a stage in development where only the first two functional requirements — the BOCA contestant interface and the PDF viewer — had been completed.

Experimental design. During the testing phase, participants were tasked with submitting solutions for four simple programming problems. The first two submissions were required to be made using the BOCA web interface, while the last two were to be completed using the BOCADE. To expedite the process, the solution for each problem

was provided alongside its description. The facilitator was readily available to address any inquiries that participants had during the testing session. For each problem, participants needed to: access the problems tab, download the problem description available as a PDF file, type the provided solution into VS Code, save it as a source code file, access the runs tab, and submit the source code file. Following the testing session, each participant answered a form containing the following three open-ended questions: (FT-Q1) "In your opinion, what are the possible improvements in the design and UX of the website?" (FT-Q2) "In your opinion, what are the possible improvements in the design and UX of the extension?" (FT-Q3) "In your opinion, which one between the website and the extension offers the better design and UX?". The form utilized in this experiment lacked a consent field, thereby compromising adherence to academic research ethics. Albeit the absence of personal data collection lessens the severity of this oversight.

Participants. The test was conducted with 15 individuals with CS backgrounds — 6 secondary education students specializing in Informatics, 6 Information Systems undergraduate students, and 3 individuals with Master's degrees in CS.

Execution. The experiment was conducted in March 2024 at the Informatics Labs of the Federal Institute Goiano — Advanced Campus Catalão.

Results. A simple qualitative content analysis of the responses was conducted. The responses to the first two survey questions revealed a preference for the BOCADE, as evidenced by participants' more critical stance towards the website. The responses to FT-Q1 revealed criticism directed towards the interface of the website, particularly regarding its outdated aesthetic design. An examination of the responses to FT-Q2 unveiled that the recommendations for the extension focused entirely on minor interface adjustments. No usability issues were found. The responses to FT-Q3 — the most pertinent question to validate the paper's contribution — revealed unanimous user preference for the extension over the web interface. Of particular note, many respondents emphasized the extension's ability to enhance user efficiency.

Discussion. The participants provided unanimously positive feedback concerning the BOCADE. Therefore, the findings substantiate our hypothesis regarding the UX improvements produced by the extension compared to the traditional setup. The participants did not identify any usability issues, and none of the minor adjustments suggested by them were judged by the authors to be significant and sufficiently appropriate to warrant implementation.

Limitations. The questionnaire had a quite limited scope. However, it sufficed for achieving the formative evaluation's objective of ascertaining initial validation. Moreover, it is important to note that the concise questionnaire did not compromise the ability to identify usability issues, as the tests were conducted individually, allowing the facilitator to closely observe users throughout the entire testing session.

5.2. Summative Evaluation

The evaluation presented in this subsection adopted a more rigorous methodology to provide a robust assessment of the extension after every planned feature had been implemented. The increased rigor was achieved by utilizing a more comprehensive questionnaire and by conducting the evaluation within the intended context of use for which the BOCADE was designed — a programming contest.

Experimental design. The testing phase took place over two hours and consisted of a two-part programming competition for students working in pairs. In each contest section, students were tasked to solve two problems. In the first section, students developed solutions in VS Code and submitted them through the BOCA web contestant interface. The subsequent section required students to code and submit their solutions entirely within the VS Code using the BOCADE extension. Following the testing phase, a five-section questionnaire was administered to gather information about students' backgrounds and experiences with both interfaces. The initial section of the questionnaire focused on collecting the email addresses of the participants and securing their consent for the use of the collected data, ensuring adherence to the norms of ethics in academic research. Thereafter, the questionnaire delved into demographics and background experiences (i.e., gender, age, interest in pursuing a CS-related undergraduate degree, and prior experience with BOCA). To assess usability, the questionnaire then employed the SUS twice — once for the BOCA web interface used in the traditional setup and again for the extension. The SUS questionnaire used was the standard version, as outlined in [Lewis 2018], and translated into Brazilian Portuguese by the authors. Lastly, the concluding section contained miscellaneous closed-ended and open-ended questions intended to further assess the extension's UX and elicit a direct comparison with the BOCA web interface's UX. These miscellaneous questions covered topics such as user perception of visual design and navigation experience for both interfaces, ranking of the extension's functionalities by usefulness, and future usage intent.

Participants. The summative usability test was conducted with sixteen final-year secondary school students specializing in Informatics, administered as an optional assignment in an introductory-level Python programming course.

Execution. The experiment was conducted in May 2024 at one of the Informatics Labs of the Federal Institute Goiano — Advanced Campus Catalão.

Results. The evaluation participants comprised 10 males and 6 females with a mean age of 17.4. In regards to academic aspirations, 4 participants expressed interest in pursuing a CS-related degree, 7 were undecided, and 4 did not intend to pursue such a degree. The sample included nine participants with prior experience in BOCA-administered programming competitions, six of whom also had previously participated in the formative evaluation. On the whole, the questionnaire's responses indicated a user preference for the extension compared to the traditional setup, albeit less pronounced than the previously observed in the formative evaluation results.

In response to a direct query regarding their preferred BOCA interface for future competitions, 14 participants indicated the extension, 1 participant preferred the web interface, and 1 participant expressed no preference. However, the SUS scores (Figure 2) indicated somewhat less favorable results for the extension. The web interface attained an mean SUS score of 51.3 (median of 52.5, F grade), whereas the extension had a mean score of 74.3 (median of 75, B grade) — a difference of 23 points between both interfaces. The individual SUS score differences had a median of 7.5 and its distribution can be divided into four groups: four participants had slightly (i.e., 10 points or less) higher scores for the web interface, one had the exact same score for both interfaces, four had slightly higher scores for the extension, and seven had significantly higher scores for the extension.

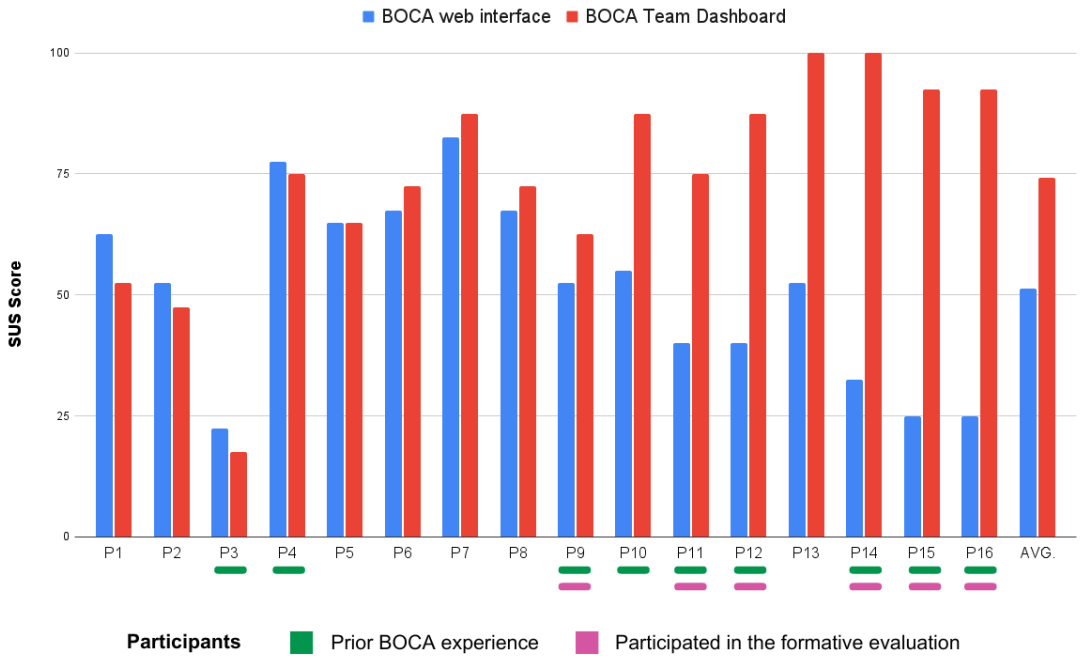

Figure 2. The SUS scores

Furthermore, the responses revealed a marked user preference for the extension's visual design, reflected in the mean scores (out of 5) of 4.31 (median of 4) compared to just 2.75 (median of 2.5) for the web interface. The results also revealed a preference for the extension in terms of navigation experience — a clear majority (10) favored the extension while 5 users reported no significant difference between the interfaces and only one user found the web interface's navigation to be more efficient. Based on user rankings of extension features from most to least useful, we calculated the mean and median (respectively) usefulness score out of 5 for each feature. The ranking were as follows: integrated PDF viewer (3.69 and 4), test cases panel (3.44 and 4.5), integrated BOCA contestant interface (2.81 and 3), organization button (2.75 and 2.5), and dark and light mode support and button (2.31 and 2). Notably, the participants did not encounter any usability issues during the evaluation.

Discussion. The analysis of responses largely aligns with our assumptions and previous results regarding the extension's positive impact on the UX. Nevertheless, the SUS scores findings present a caveat. The substantial discrepancy between the mean and median of the SUS score differences (23 vs 7.5) garnered our attention, and, therefore, we conducted a through analysis of the responses to explore the reasons behind it. The analysis revealed a relationship between SUS score difference and participation in the formative evaluation. Participants in this groups exhibited a much higher mean SUS score difference (49) than participants who did not took part in the formative evaluation (7). Particularly noteworthy is the very high mean SUS score of 85 (median of 90, A+ grade) attributed to the extension by the participants who had taken part in the previous evaluation. Our hypothesis for such a significant disparity is that these participants likely benefited from a learning effect due to their prior exposure to the extension. This familiarity presumably led to reduce cognitive load and more efficient use. Another related

factor that could have contributed to the disparity is that the extension may not have been adequately introduced to the students. Due to the number of users participating in the test simultaneously, the sole facilitator was unable to provide individualized guidance to the same extent as in the formative evaluation.

Furthermore, the responses to the open-ended questions were quite concise, merely reinforcing the closed-ended responses without offering further information and, therefore, did not yield any actionable insights. On the other hand, the significant difference in SUS scores based on user familiarity signaled the importance of incorporating a help section to provide basic guidance on utilizing the extension. The BOCA web interface lacks any help-related features; therefore, the inclusion of a help section would be yet another improvement provided by the extension.

Threat to validity. The learning effect observed in the SUS score differences among participants with prior exposure to the extension suggests that the extension's perceived usability may not be as significantly superior as the overall 23-point difference might imply. The more modest 7-point difference among participants without prior experience seems to indicate that the extension's overall superiority in usability is less pronounced when evaluated by first-time users. However, it is necessary to account for the possibility that the less favorable usability ratings from first-time users stems from insufficient guidance during the testing. Moreover, the markedly higher rating for the extension from the group with prior experience (35.9 vs. 85) is very favorable for the extension, particularly because these users were familiar with both interfaces and therefore could provide a better informed comparison, irrespective of any potential instructional shortcomings during testing.

6. Conclusions and Future Work

This paper presented BOCADE, a VS Code extension that optimizes the code editor for use in BOCA-administered programming competitions. To assess the effectiveness of this novel contribution, we conducted two usability tests to evaluate the users' overall experience with the system: a formative and a summative test. The findings indicated that the extension offered superior usability and aesthetics — two crucial aspects of UX — compared to the traditional setup. Although the evaluations were subject to limitations, specifically the restricted scope of the formative evaluation questionnaire and the learning effect identified in the summative evaluation results, the overall findings provide reasonable evidence to support the conclusion of improved UX.

In regards to future work, this paper is part of a broader work undertaken by the authors, with the overarching goal of augmenting the BOCA system to enhance the UX for both contestants and organizers of programming competitions. The present work has focused on our contributions towards an improved UX for contestants. In subsequent work, we intend to direct our research efforts towards the enhancement of organizers' UX and propose software to that effect.

Acknowledgements

This paper is a result of the research of the scientific initiation conducted with the support of the Institutional Program for Scientific and Technological Initiation Scholarships (PIBIC) subsided by the Federal Institute Goiano.

The authors acknowledge the use of generative AI tools, specifically ChatGPT 4o and Gemini 1.5 Flash, as revision aids in the writing of this paper.

Data Availability Statement

The data that support the findings of this study are openly available here. BOCADE is available as open-source software here.

References

Adithya, V. C. (2011). *Security sandboxing for PC^ 2*. PhD thesis, California State University, Sacramento.

Agrawal, D. (2020a). Competetive Programming Helper. https://github.com/agrawal-d/cph.

Agrawal, V. (2020b). compile-run. https://github.com/vibhor1997a/compile-run/.

Algar Telecom (2012). Roteiro para configuração de minimaratona de programação. https://docplayer.com.br/36306096-Minimaratonas-de-programacao.html.

Audrito, G., Ciobanu, M., Laura, L., et al. (2023). Giochi di fibonacci: Competitive programming for young students. *Olympiads in Informatics*, pages 19–31.

bin Uzayr, S. (2022). *Mastering Visual Studio Code: A Beginner's Guide*. CRC Press.

Cheah, C. S. (2020). Factors contributing to the difficulties in teaching and learning of computer programming: A literature review. *Contemporary Educational Technology*, 12(2):ep272.

Combéfis, S. and Wautelet, J. (2014). Programming trainings and informatics teaching through online contests. *Olympiads in Informatics*, 8.

Coore, D. and Fokum, D. (2019). Facilitating course assessment with a competitive programming platform. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 449–455.

de Campos, C. P. and Ferreira, C. E. (2004). BOCA: um sistema de apoio a competições de programação. In *Workshop de Educação em Computação*. Sociedade Brasileira de Computação.

do Carmo Nogueira, T., de Souza Campos, E., and Ferreira, D. J. (2018). Cognition developing of computer higher education students through gamification in the algorithm teaching-learning process. In *Anais do XXVI Workshop sobre Educação em Computação*. SBC.

DOMjudge (2024). DOMjudge – About the project. https://www.domjudge.org/about.

dos Santos, L. G. A. (2024). BOCA: funcionalidades. https://github.com/gusalbukrk/boca/tree/main/funcionalidades.

Flórez, F. B., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., and Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, 87(4):834–860.

Ganorkar, C. S. (2017). PC2 web team interface. *ScholarWorks*.

Glez-Peña, D., Lourenço, A., López-Fernández, H., Reboiro-Jato, M., and Fdez-Riverola, F. (2014). Web scraping technologies in an API world. *Briefings in bioinformatics*, 15(5):788–797.

ICPC Foundation (2024). The ICPC International Collegiate Programming Contest. https://icpc.global.

International Organization for Standardization (ISO) (2018). Ergonomics of human-system interaction—part 11: Usability: Definitions and concepts (iso 9241-11: 2018).

jsdom (2024). jsdom. https://github.com/jsdom/jsdom.

Kostadinov, B., Jovanov, M., and Stankov, E. (2010). A new design of a system for contest management and grading in informatics competitions. *ICT Innovations Conference 2010*.

Kurtanović, Z. and Maalej, W. (2017). Automatically classifying functional and non-functional requirements using supervised machine learning. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 490–495. Ieee.

Lewis, J. R. (2018). The system usability scale: Past, present, and future. *International Journal of Human–Computer Interaction*, 34(7):577–590.

Lima, A. L. d. S. and Benitti, F. B. V. (2021). Usabilityzero: Can a bad user experience teach well?. *Informatics in Education*, 20(1):69–84.

Lima, D. T., Moura, F. R. T., Alves, A. V. N., de Moura Parracho, T., Zacarias, R. O., dos Santos, R. P., and da Rocha Seruffo, M. C. (2022). Ux-tracking: Web and multimodal tool for user experience evaluation. In *Anais Estendidos do XXVIII Simpósio Brasileiro de Sistemas Multimídia e Web*, pages 107–110. SBC.

Maggiolo, S. and Mascellani, G. (2012). Introducing CMS: a contest management system. *Olympiads in Informatics*, 6:86–99.

Majumdar, A. (2017). MacACM: Encouraging competitive programming via mentorship and outreach. *XRDS: Crossroads, The ACM Magazine for Students*, 24(1):14–15.

Meta Open Source (2024). React. https://react.dev/.

Microsoft (2024a). TypeScript. https://www.typescriptlang.org/.

Microsoft (2024b). Visual Studio Code. https://code.visualstudio.com/.

Microsoft (2024c). Webview UI Toolkit for Visual Studio Code. https://github.com/microsoft/vscode-webview-ui-toolkit/.

Morais, W. B. and Ribas, B. C. (2019). Maratona-Linux: um ambiente para a Maratona de Programação. *Anais do Computer on the Beach*, pages 416–426.

Moreno, J. and Pineda, A. F. (2018). Competitive programming and gamification as strategy to engage students in computer science courses. *Revista ESPACIOS*, 39(35).

Murty, R. N., Dadlani, S., and Das, R. B. (2022). How much time and energy do we waste toggling between applications? *Harvard Business Review*. https://hbr.org/2022/08/how-much-time-and-energy-do-we-waste-toggling-between-applications.

Pham, M. T. and Nguyen, T. B. (2019). The DOMJudge based online judge system with plagiarism detection. In *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*, pages 1–6. IEEE.

Piekarski, A. E., Miazaki, M., Hild, T., Mulati, M. H., and Kikuti, D. (2015). A metodologia das maratonas de programação em um projeto de extensão: um relato de experiência. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 4, page 1246.

Riihiaho, S. (2018). Usability testing. *The Wiley handbook of human computer interaction*, 1:255–275.

Rodrigues, L. and Isotani, S. (2023). When personalized gamification meets computing education: A multidimensional approach to motivate students to learn. In *Anais do XXXVI Concurso de Teses e Dissertações*, pages 50–59. SBC.

Sentance, S. and Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and information technologies*, 22:469–495.

Silva, T. R. d. M. B., Braga, G., Silva, M. A. L., Araújo, M., et al. (2023). Maratonando! inspirando e capacitando programadores com diversidade de gênero e variedade de competições. In *Anais do XVII Women in Information Technology*, pages 346–351. SBC.

Singer-Vine, J. (2024). pdfplumber. https://github.com/jsvine/pdfplumber.

Stack Overflow (2023). Stack overflow developer survey 2023. https://survey.stackoverflow.co/2023/#section-most-popular-technologies-integrated-development-environment.

Tenório, T. and Bittencourt, I. I. (2016). A gamified peer assessment model for on-line learning environments: An experiment with MeuTutor. In *Anais do XXIX Concurso de Teses e Dissertações*, pages 381–386. SBC.

Tomoki, M. (2023). vscode-pdf. https://github.com/tomoki1207/vscode-pdfviewer.

Verma, R. (2020). *Visual Studio Extensibility Development*. Springer.

Xia, B. S. (2017). A pedagogical review of programming education research: what have we learned. *International Journal of Online Pedagogy and Course Design (IJOPCD)*, 7(1):33–42.

Yuen, K. K., Liu, D. Y., and Leong, H. V. (2023). Competitive programming in computational thinking and problem solving education. *Computer Applications in Engineering Education*, 31(4):850–866.

Zhan, Z., He, L., Tong, Y., Liang, X., Guo, S., and Lan, X. (2022). The effectiveness of gamification in programming education: Evidence from a meta-analysis. *Computers and Education: Artificial Intelligence*, 3:100096.