Design and evaluation of a scaffolded block-based learning environment for hierarchical data structures

Pedro Guillermo Feijóo-García [1,2], Sishun Wang[1], Ju Cai[1], Naga Polavarapu[1],
Christina Gardner-McCune[1], and Eric D. Ragan[1]
Department of Computer & Information Science & Engineering[1]
University of Florida, Gainesville, FL, U.S.A.[1]
Program of Systems Engineering[2]
Universidad El Bosque, Bogotá, Colombia[2]
Email: {pfeijoogarcia, sishunw, jucai1, npolavarapu, gmccune, eragan}@ufl.edu

***Abstract*— This paper presents the design of Blocks4DS, a block-based environment for students to learn data structures. As a proof-of-concept, we designed custom blocks to allow students to build and visualize Binary Search Trees (BST). Blocks4DS is built on Blockly and uses vis.js to provide visualizations of the binary search tree and its operations. This paper describes the results from an initial evaluation of usability and student learning.**

***Keywords*—CS Education, tools, block-based programming, data structures, visual languages, human-centric computing**

I. INTRODUCTION

Data structures courses present levels of abstraction that make them challenging for students to learn and for instructors to teach. These courses generally correspond to the second year of Computer Science (CS) curricula, in which students usually address their concepts through programming languages that come with syntax and language complexity even though the data structure topics can be addressed without them. These courses pose pedagogical challenges that require both the instructor and the student to invest effort into addressing the different levels of abstraction in designing and organizing data structures implementation [1, 2].

We designed *Blocks4DS,* an instructional block-based technology for students to learn data structures without requiring the extra difficulties of programming languages, syntax, and configuration constraints. The initial modules support the design and development of the Binary Search Tree (BST), offering the opportunity for students to build, dynamically debug, and visualize the data structure. In this paper, we describe the design of *Blocks4DS*, also presenting findings and results on usability and learning.

II. BACKGROUND

Recent CS ED studies have explored data structures courses pedagogically and through a curricular lens. Porter et al. [2] found that CS2 courses across multiple institutions were going beyond teaching object-oriented programming, regardless of the standards, by introducing basic data structures (from arrays to the binary search tree). Open questions include whether data structures should be introduced to students immediately after the introduction to programming. Do data structures topics imply the same complexity as introductory programming concepts? The decision of introducing data structures topics in the first year (CS1 or CS2) can potentially impact learning outcomes due to possible difficulties learners might face, as Zingaro et al. [1] found while examining students' difficulties with basic linear and hierarchical data structures such as ArrayLists, simple and doubly linked lists, and Binary Search Trees (BST). The authors found that most difficulties related to the BST, considering aspects such as data insertion, searching, and algorithmic complexity. They also encourage the CS ED community to further explore this topic, arguing that literature on data structures education is still limited.

Over the past decade, different tools have been developed to scaffold students' learning data structures. Some offer visual-based languages to reduce difficulties regarding syntax, while others feature visualizations and simulations to provide students with visual feedback of their solutions.

Leveraging the Benefits of Block-based Environments for Teaching Programming. Visual programming environments such as *Scratch*, *Blockly*, and *Alice* provide students with programming block-constructs that snap together and can be combined to create programs without worrying about syntax. Despite their simplified visual presentation, and drag-and-drop affordances, these blocks can be sequenced and nested into complex programs that solve real programming challenges. Moreover, these programming environments have been used successfully by novice programmers to learn fundamental computing concepts [3, 4]. Price & Barnes [3] found that block-based environments can help novices perform better when assessing programming activities, mainly in reducing the time they require to complete them. Similarly, Feijóo-García & De la Rosa [5] found that students perceive block-based environments friendly and interesting, based on a study they conducted with middle school students who used *Scratch* and *RoBlock*.

As block-based programming environments mature, they are beginning to offer hybrid features that allow students to toggle between blocks and text representations. Weintrop & Holbert [6], reporting on the advantages and drawbacks of block-based and text-based programming languages, found that the block modality was most commonly used by learners to

978-1-7281-0810-0/19/$31.00 ©2019 IEEE

explore and discover new commands. Working with the tool *Tiled, Grace*, Homer & Noble [4] noted that participants found the tiled view useful to explore and that switching between tiled and textual views helped them to better understand concepts. Block-based programming environments help learners explore CS concepts before tackling them in text, and have led to increased understanding and confidence when using concepts. Although generally used for introductory courses such as CS1, and for K-12, block-based technologies can be proposed and used for courses such as data structures, helping the students to visualize, build and debug graphically.

Block-based Environments for Teaching Data Structures. More recently, block-based and hybrid programming environments are being developed to address the challenges students face in learning data structures. Almanza-Cortés *et al.* [7] evaluated the use of block-based technologies to assist in understanding linear data structures. Although limited to linked-lists, stacks, and queues, *DStBlocks* offers an environment that allows the student to visualize the data structures while coding and building them [7]. Their work demonstrates meaningful findings and results on the impact of block-based technologies for simple data structures and encouraged us to explore the use of block-based technologies for hierarchical data structures instruction.

Data Structures Visualization Tools. In addition to tools to help students program data structures, another set of tools helps them visualize the manipulations of the actual data structure. One example is *BRIDGES*, which is a system to enable the creation of engaging data structures assignments with real-world data and visualizations [8]. Burlinson et al. [8] found that *BRIDGES* could provide an easy-to-use way for accessing interesting real-world datasets, allowing the students to share the visualization of data structure with others. Taking a similar approach, Halim [9] presents *VisuAlgo*, as a scaffolded tool to visualize different data structures and algorithms. *VisuAlgo* offers visualizations for binary and non-binary trees, including pseudo-code tracking for CRUD operations (i.e., Create-Read-Update-Delete). Visualizing data structures helps students to understand not only the structures' design but also their functionality. However, visualization-only tools provide pre-built data structures and do not provide mechanisms to help students build them by themselves. As a result, students are not able to see connections between the data structures and the code required for them, which is something block-based technologies can offer.

III. BLOCKS4DS: OUR SOLUTION

Blocks4DS is a block-based environment designed to help students learn about data structures (http://b4dsprototype.herokuapp.com/). As a proof-of-concept, we designed custom blocks to allow students to build and visualize Binary Search Trees (BST) and their operations. In this section, we describe the technologies used for its development, presenting its design and its features.

A. Technologies Used

Blocks4DS was developed using a series of libraries, languages, and components for web development. At its core, the tool was built using JavaScript structured with HTML and styled with CSS. For more specialized functionality, we used *Blockly* and *vis.js*. We used *Blockly*, an open source library that supports authoring visual block-based code environments [10]. It provides functionality for code translations, which are useful when showing students how their blocks correspond to conventional languages such as JavaScript and Python [10]. Additionally, we used *vis.js*, which is a web library for visual representation of data [11]. This library was incorporated in our solution to allow visualizing the BST and its functionality according to their code (Figure 1).

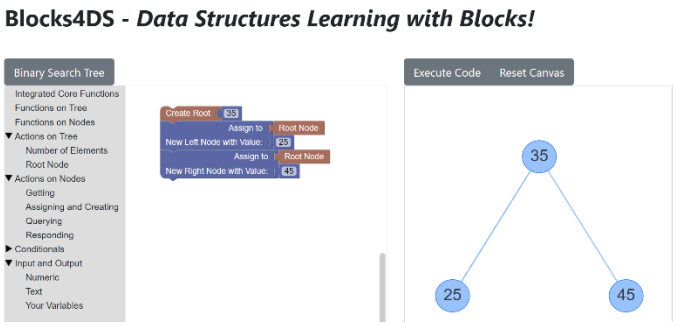

Figure 1. Graphical visualization of BST

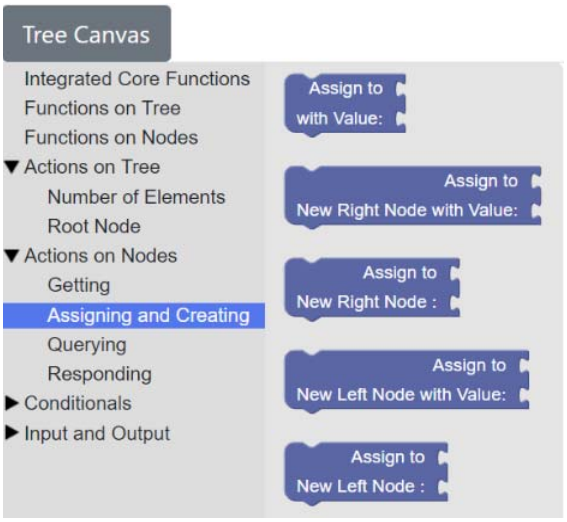

Figure 2. Categories and Sub-Categories

B. Scaffolding Strategies & Blocks4DS Features

The blocks in *Blocks4DS* are visually and conceptually organized into seven different categories based on their intended use (Figure 2). The first category, *Integrated Core Functions*, offers action-blocks for CRUD (i.e., Create-Read-Update-Delete) operations in the BST: insertion, deletion, and search. These blocks are included so the student can quickly build a BST and use it to test CRUD functions without having to implement them all (e.g., insertion for search). The first block we implemented in this category is *Insert Element.*

With the blocks-programming visual editor, students can complement the functionality of a BST, which corresponds to two classes: Tree and Node (Figure 3). The *Functions on Tree* and *Functions on Node* are the two categories that gather the

core procedure-blocks referring to the main CRUD actions of the data structure. Also, with the assistance of multiple built-in functions of the BST, like *insertElement*, *searchElement*, or *removeElement*; the application can be used by the student to visually construct the BTS (Figure 4).

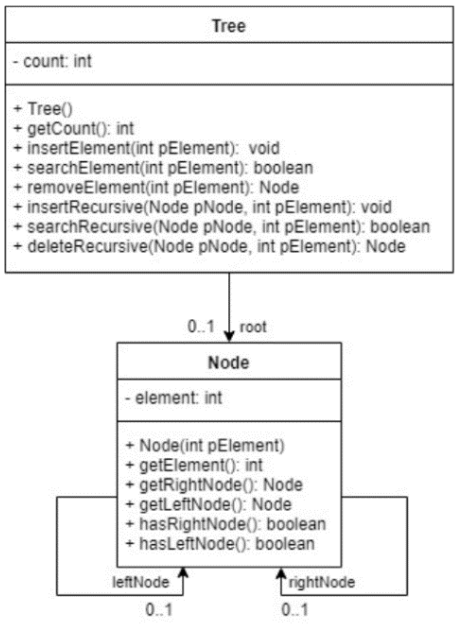

Figure 3. BST - UML Class Diagram

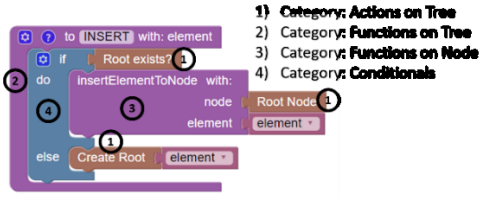

Figure 4. insertElement with Blocks

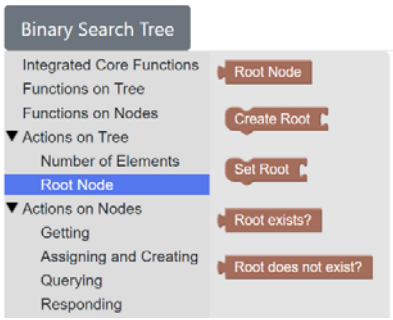

Figure 5. Sub-category: Actions on Tree

The next two categories, *Actions on Tree* and *Actions on Node*, present a series of action-blocks corresponding to setters and getters of the properties and associations per class (Figure 2). All actions within these categories are displayed ordered by type-signified sub-categories, such as *Root Node*, *Getting, Querying, Assigning and Creating*, among others (Figure 5).

The final two categories, *Conditionals* and *Input and Output*, present blocks that correspond to selection instructions (i.e., if-else statements) and to data request/response for numerical and textual values.

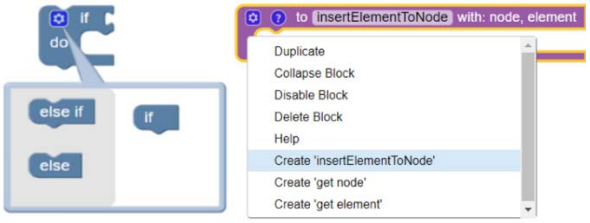

Figure 6. Extra Features

Most sub-categories blocks can display extra features and elements by right-clicking on those blocks or by left-clicking on the gear icon displayed within those blocks (Figure 6). Additionally, all blocks in a same sub-category are presented in the same color to help ease selection and support interaction-memory.

IV. Preliminary Evaluation

We conducted a preliminary evaluation of Blocks4DS with a focus on learning and user-experience outcomes. To help study the tool's impact, we designed and implemented a video-lecture and assignment intended to introduce participants to BSTs.

Participants. We piloted our study with seven participants, all undergraduate CS-major freshmen university students (n = 7). Participants were male students between 18 to 23 years of age, recruited from the COP 3503 Programming Fundamentals II course at the University of Florida. Unfortunately, we were unable to recruit female students to participate in this pilot study.

Experiment Description. We met with each participant for 60 min. Participants were first asked to complete a pre-assessment: four questions asking which visual examples were a BST, and one open-ended question to explain how to search for an element in a BST. They were then asked to watch a 20 min step-by-step video introducing the BST concept and to complete an assignment that asked them to code the *Search* function of the BST using (https://bit.ly/324Iz6T). After completing the assignment, participants completed a post-assessment with the same structure of the pre-assessment. Finally, participants were asked to complete a final questionnaire with Likert 1-7 scale and open-ended questions about their experience.

Data Collection & Analysis. We evaluated the learning-outcomes using pre- and post-assessments to measure participants' conceptual understanding of BST before and after interacting with the tool. Both assessments featured quantitative and qualitative questions that asked students to select correct examples of the BST, and to explain the procedure of searching for an element in the data structure. Students' performance outcomes in both assessments were based on the number of correct responses to the quantitative questions, and to the details' accuracy provided in the open-ended search procedure question. We evaluated the quality of details using a Likert-scale rubric (1-*no idea*, 4-*has some idea*, and 7-*explained completely*), used by the research team to grade each open-ended response.

For the user-experience perspective, we measured the time participants spent to complete the video-assignment in addition to the students' perceptions gathered through a post-survey. Responses were gathered with Likert scales addressing aspects such as Graphical User Interface (GUI) organization, difficulty of the assessment, and difficulty using the tool. Additionally, we obtained textual feedback through an open-ended question included in the questionnaire.

Our evaluation sought to establish evidence for *Blocks4DS* functionality and feasibility, as well as to gather and understand the feedback from the participants. Descriptive statistics were used to analyze the data we present in Section V.

V. RESULTS AND DISCUSSION

A. Pilot Study – UX Perspective: Findings and Results

All participants were able to successfully complete the tasks required with the video tutorial. The average time of completion of the assignment was below to half an hour $[Mean = 27\ min, SD = 7\ min]$.

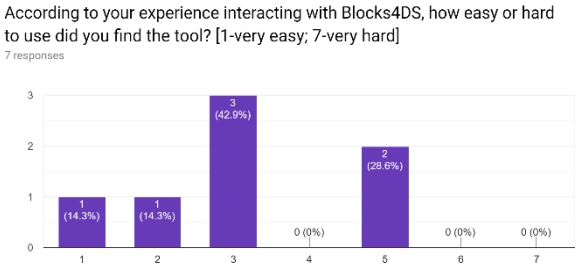

Figure 7. How easy or difficult was to use the tool?

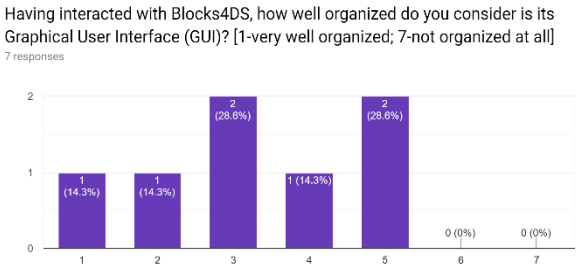

Figure 8. How well organized do you consider is its GUI?

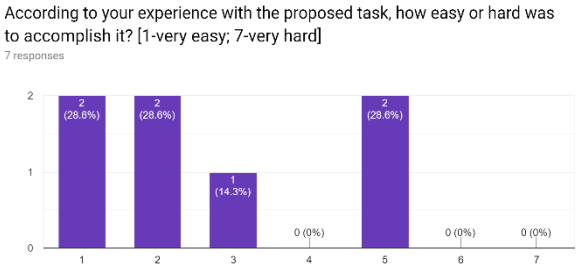

Figure 9. How easy or hard was the assignment to accomplish?

Five participants out of seven self-reported that the tool was "easy" to use (Figure 7). Four considered the GUI well organized (Figure 8), and five responded that the proposed assignment was easy to accomplish (Figure 9). These responses suggest that the first pilot was positively received by the intended audience.

B. Pilot Study – Learning-Outcomes: Findings and Results

The responses' distribution per type of question are presented in Figure 10 (Quantitative) and Figure 11 (Qualitative). In general, as we can observe, participants improved their understanding of BST for the post-assessment.

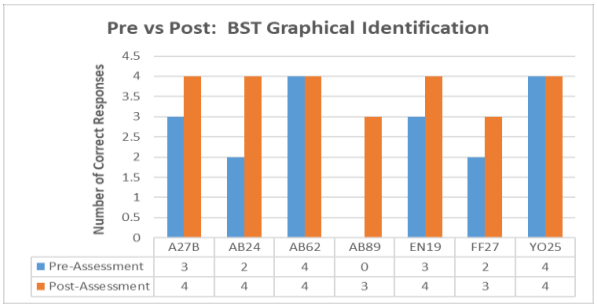

Figure 10. Quantitative Responses

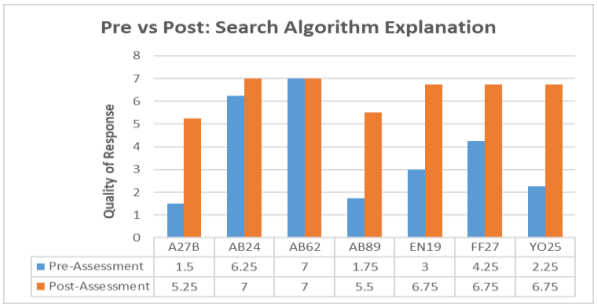

Figure 11. Qualitative Responses

On average, for both types of questions, participants performed better after interacting with the learning tool. Additionally, our results present lower standard deviations for the post-assessment:

- Quantitative - Pre vs Post:
  - Pre: $[Mean = 2.57,\ SD = 1.39]$.
  - Post: $[Mean = 3.71,\ SD = 0.49]$.
- Qualitative - Pre vs Post:
  - Pre: $[Mean = 3.71,\ SD = 2.19]$.
  - Post: $[Mean = 6.42,\ SD = 0.73]$.

VI. DISCUSSION, CONCLUSIONS, AND FUTURE WORK

Based on our experimental method and results, we conclude that *Blocks4DS* can be used to help students learn basic concepts of data structures in a short amount of time (< 60 min). Additionally, considering our findings and participant feedback, we conclude that *Blocks4DS*' graphical organization is appropriate for the learning context it was designed for.

Further research with the tool is necessary to compare the usage of the tool with a traditional text-based language. In subsequent studies, we will explore how impactful the tool can be when used for comparing different concepts, lectures/video-tutorial, and assignments.

REFERENCES

[1] D. Zingaro, C. Taylor, L. Porter, M. Clancy, C. Lee, S. N. Liao, and K. C. Webb, "Identifying Student Difficulties with Basic Data Structures," Proceedings of the 2018 ACM Conference on International Computing Education Research - ICER 18, 2018.

[2] L. Porter, D. Zingaro, C. Lee, C. Taylor, K. C. Webb, and M. Clancy, "Developing Course-Level Learning Goals for Basic Data Structures in CS2," *Proceedings of the 49th ACM Technical Symposium on Computer Science Education - SIGCSE 18*, 2018.

[3] T. W. Price and T. Barnes, "Comparing Textual and Block Interfaces in a Novice Programming Environment," Proceedings of the eleventh annual International Conference on International Computing Education Research - ICER 15, 2015.

[4] M. Homer and J. Noble, "Combining Tiled and Textual Views of Code," 2014 Second IEEE Working Conference on Software Visualization, 2014.

[5] P. G. Feijóo-García and F. De la Rosa, "RoBlock – Web App for Programming Learning," International Journal of Emerging Technologies in Learning (iJET), vol. 11, no. 12, p. 45, 2016.

[6] D. Weintrop and N. Holbert, "From Blocks to Text and Back: Programming patterns in a dual-modality environment " Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE 17, 2017.

[7] D. F. Almanza-Cortés, M. F. Del Toro-Salazar, R. A. Urrego-Arias, P. G. Feijóo-García, and F. D. L. Rosa-Rosero, "Scaffolded Block-based Instructional Tool for Linear Data Structures: A Constructivist Design to Ease Data Structures' Understanding," International Journal of Emerging Technologies in Learning (iJET), vol. 14, no. 10, p. 161, 2019.

[8] D. Burlinson, M. Mehedint, C. Grafer, K. Subramanian, J. Payton, P. Goolkasian, M. Youngblood, and R. Kosara, "Bridges, A system to enable creation of engaging data structures assignments with real-world data and visualizations" Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE 16, 2016.

[9] S. Halim, "VisuAlgo – Visualising Data Structures and Algorithms Through Animation," *Olympiads In Informatics*, vol. 9, pp. 243–245, 2015.

[10] N. Fraser, "Blockly Games," *Blockly Games*. [Online]. Available: https://blockly-games.appspot.com/. [Accessed: 21-Mar-2019].

[11] B. V. Almende, "vis.js," vis.js - A dynamic, browser based visualization library. [Online]. Available: http://visjs.org/. [Accessed: 21-Mar-2019].