UDC 377.5
DOI: 10.23951/2782-2575-2023-2-60-76

THE EFFECTIVENESS OF USING A CONGRUENT VISUALIZATION FRAMEWORK ON LEARNING A DATA STRUCTURES COURSE

Adam Basigie Mtaho[1], Leonard J. Mselle[2], Masoud M. Masoud[3]

[1] *Arusha Technical College, Tanzania, abasigie@.yahoo.com*

[2] *The University of Dodoma, Dodoma, Tanzania, mselel@yahoo.com*

[3] *Dar es Salaam Institute of Technology, Dar es Salaam, Tanzania, bi-gutu@gmail.com*

Abstract. The majority of computer science (CS) educationists agree that learning the Data Structures course (CS2) is very difficult among novices due to its complexity. Consequently, learning the Data Structures course has been associated with a high failure rate. To enable learners understand data structures, algorithms visualizations (AVs) were proposed. Despite the long-term use of AVs in teaching and learning data structures, research shows that such tools have not been as pedagogically effective as expected. This study aimed to evaluate the effectiveness of using a congruent visualization (CV) framework on learning data structures. The framework employs a combination of two congruent program visualization tools, which involve machine-driven and learner-driven approaches. The effectiveness of using the CV framework was evaluated using a combination of experiment, grade analysis, and questionnaire methods. The subjects of the study were 887 first-year undergraduate students from the College of Informatics and Virtual Education (CIVE) of the University of Dodoma in Tanzania, studying the CS 122 Data Structures course. Results show that the use of the CV framework improved both students' test performance and examination pass rates compared to the traditional approach. Students' responses from a follow-up survey showed that the use of the CV framework increased students' motivation and confidence in learning the Data Structures course.

Keywords: *Data Structures course, Visualization, Program visualization, Algorithm Visualization, Congruent visualization framework*

For citation: Mtaho A.B., Mselle L.J., Masoud M.M. The Effectiveness of Using a Congruent Visualization Framework on Learning a Data Structures Course. *Education & Pedagogy Journal*. 2023;2(6):60-76. doi: 10.23951/2782-2575-2023-2-60-76

Introduction

Data structures is one of the prerequisite courses in CS education. Data Structures course is crucial as it enables students to apply computer programming skills to solve real-life problems. However, the subject has been associated with high failure rates due to its abstract and dynamic nature [1–4]. The authors in [2] report of high failure rate in data structures ranging from 30%-40% among CS undergraduate students. In another study, the authors in [5] report the students' dropout rates of over 70%. Currently, teaching data structures in most universities and colleges worldwide is done through the traditional approach based on lectures, tutorials, and lab sessions along with algorithms visualization (AV) tools [6–8]. This approach is rife with the split attention effect because the content, working environments, and demonstrating tools are separate, leading to a high extraneous cognitive load [9]. There are also difficulties due to the mismatch between the illustrations provided in the textbook and those provided by the instructors and AV tools [10]. In addition, the use of static images/diagrams in several data structures textbooks to convey a dynamic process has little effect on reducing students' cognitive load [6]. The major question that has preoccupied researchers in CS education is how to reduce the cognitive load in learning the Data Structures course in order to increase comprehension and ultimately reduce the failure rates.

Numerous studies have shown that the use of AV in teaching and learning data structures can help improve data structures understanding for students and ultimately improves comprehension [11, 12]. However, despite the claimed educational benefits of using AVs in teaching and learning data structures, studies show that the sole use of AVs for teaching data structures is not pedagogically effective in learning data structures for two main reasons; Firstly, such tools are still incompatible with standard textbooks used for teaching data structures and algorithms [10][13]; and secondly, these AVs have focused mainly on demonstrating algorithmic steps instead of showing the programming logic behind those steps while manipulating data structures elements [10, 14–16]. In order to address this deficiency, this study introduces a new congruent visualization (CV) framework that combines the use of two congruent program visualization (PV) tools, one of which is machine-driven while the other is learner-driven, working in tandem. The proposed framework works as AV, PV, and compiler. The rest of this paper is organized as follows: It starts with the literature review; then the overview of the CV framework, followed by the study methodology. Lastly, it presents the results, discussion, and the conclusion.

Literature review

Visualization Overview

According to the authors in [17], visualization refers to the "use of graphical representation of information to assist human comprehension of, and reasoning about, that information." Visualization in programming is grouped into two fundamental categories, namely, software visualizations and manual or learner-driven visualizations. Software visualization applies "the use of the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction technology to facilitate both the human understanding and effective use of computer software" [18]. Software visualization is further classified into three main categories, which are AV, Program Visualization (PV), and Visual programming language (VPL).

PV visually shows the program state, its execution, and data structures when the program runs inside the computer memory. AVs, on the other hand, are intended to help learners clearly understand conceptually how the algorithm works but do not go as far as illustrating how each program component executes when the program runs in the memory. For this reason, AVs have mostly been used to complement lectures. According to the author in [19], AVs are easier to follow than textbooks and are alternative representations for visual learners who understand materials in a visual or graphical format more quickly than in text format. Unlike AVs and PVs, VPL programs use visual images to help learners develop a computer program (Myers, 1990). VPLs are also known as "visual languages" or "visual programming." Examples of PVs, AVs, and VPLs are Balsa [20], Scratch [21], and Jeliot 3 [22], respectively. PVs and AVs are more helpful for both instructors and students; since they improve teaching efficiency; and help students understand programming logic more easily and efficiently by presenting learning materials graphically compared to textually [23].

PVs and AVs are subdivided into; dynamic, interactive, static, learner-driven, and machine-driven [24, 25]. In dynamic visualization, the visual image is updated whenever the line of code or program data executes. An interactive visualization provides an environment in which a user can manage and manipulate the visualization behavior of a program, such as stopping, forwarding, and controlling how the program or parameters should be executed and displayed. It also enables users to input data and control the visualization process. Interactive animation differs from computer animation since it is not interactive. Such a program is intended for "passive viewers "of the computer programs [26]. In static visualization, the displayed visual image just appears in a single view and does not change with time. Printed and soft images or sketches are examples of static visualization [26]. Finally, in learner-driven or manual visualization, learners use their own hands to draw the visual images on paper to show how the program execution occurs in a notional machine [13, 25, 27]. This visualization can work as AV or PV, depending on their construction. Unlike traditional static diagrams, learner-driven visualization provides full control to the learner and promotes the learner's computational thinking, confidence, and problem-solving skills [25]. Typical examples of learner-driven visualization tools are; Memory Transfer Language (MTL) [25] and Program Working Storage (PWS) [13].

The Need to Use Complementary Visualization in Teaching and Learning

Using more than one type of visual support learning is common in computer science education. The benefit of using a complementary visualization approach has been explained by Buckhard Knowledge Visualization Model [33]). Figure 1 shows the Burkhard knowledge visualization model [33].

The model describes inter- and intrapersonal iterative processes: The sender initiates the knowledge transfer process by imparting some of his knowledge (knowledge) to a recipient [33]. The knowledge of the senders as perceived by him (mental model sender) will be externalized into various explicit and complementary visual illustrations and can be divided into three sub-processes (1, 2, 3) following a temporal sequence: First, the sender must ensure the attention (1) of the leaner (recipient), for instance by using a provocative image. Second, the sender will illustrate the context (2), provide an overview (2), and then provide alternative ways to accomplish the task (act) (2). Only then can the sender point to selected details (3), which ideally happens in a dynamic dialog with the recipient (D), who re-constructs (C) similar knowledge (knowledge') with these complementary visualizations and an own mental image (mental model recipient). However, due to different assumptions, beliefs, or backgrounds, inferences can occur, and this leads to interruptions and failures in knowledge reconstruction (E). If the receiver is aware of the misinterpretation or misunderstanding or simply desires more information, questions may arise, and a feedback loop may be initiated. The sender then has to refine or modify the used visual representations, add further complementary visualizations, or use other formats to achieve a concise reconstruction of the knowledge as intended and thus make successful communication possible (F) [33].

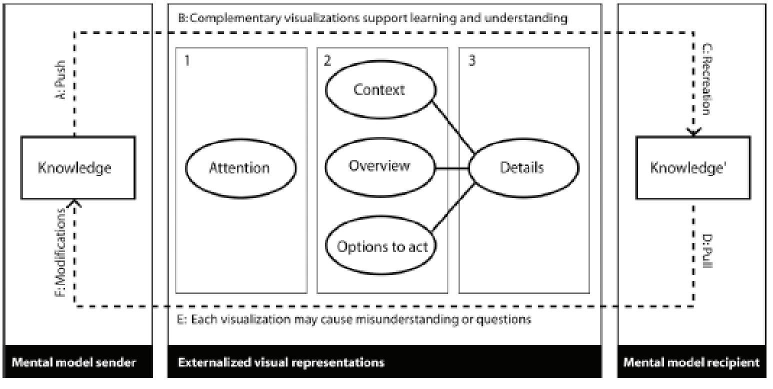

Fig. 1. Knowledge Visualization Model [33]

Related Works

Several studies have examined the impact of using visualizations in learning data structures. Some of the few works which are closely related to our works are those of the authors in [28–30]. The authors in [28] conducted an experimental study to compare the effect of using text-based instructional materials, interactive software visualization, or their combination on learning data structures. Results showed that using a combination of text and interactive visualizations improved knowledge retention and performance compared to using either text or visualization in isolation.

The authors in [29] studied the impact of using a PV tool called Courseware in learning data structures. Results showed that students who used the Courseware outperformed those who used the traditional lecture approach. Another experiment conducted by [30] to evaluate a tool called DS-PITON which combines both PV and AV functionalities in one package, showed that using DS-PITON improved students' performance compared to those who used the traditional approach.

According to the authors in [31], most of the existing PV systems have been built with less interactive and engaging pedagogical features for novice learners. Similar results have also been reported by the author in [32]. Several studies report the use of PV in teaching and learning data structures to improve comprehension. However, none of these studies employed congruent PVs. Likewise, The insights highlighted in Bukhard knowledge Visualization model are essential if applied in teaching and learning programming [33]. Currently, there is no well-known study that has investigated the effectiveness of using complementary visualization tools in learning programming. In this study, MTL, a PV tool based on learner-driven instructional materials that also works as an animator and compiler, is employed along with a new machine-driven PV tool called CeliotM. The two PVs are congruent to one another and are used in tandem in teaching and learning data structures. The study hypothesizes that using the framework, which employs two visualization tools, of which one is machine-driven, and the other is learner-driven, which are congruent and complementary to each other, will help improve students' motivation and understanding in learning the Data Structures course.

An Overview of the Congruent Visualization Framework

The CV framework is an instructional approach intended for teaching introductory programming and Data Structures courses. The approach consists of two congruent components, the learner-driven, and the machine-driven components. Being congruent implies that images used in the learner-driven approach are similar to those used in the machine-driven approach. The two approaches (learner-driven and machine-driven) are mutually reinforcing, and their order of application is inconsequential. This implies a combination of a single manual approach and a similar machine-driven animation with embedded questions so that learners can perceive the view, construct the view, respond to questions, and construct the visualization as desired.

The Learner-driven Component

The learner-driven components comprise the instructional materials prepared in MTL visual format [25]. Such materials are congruent with machine-driven visualizations. The CV framework combines human computational thinking, manual and machine-driven compilation, and visualizations. When solving a programming problem under the CV framework, a novice programmer has to read the written code in a high-level language line-by-line visually using memory diagrams. Using MTL, novices can visualize program behavior by showing the memory status for each line execution. Figure 2 shows a simple code, *Program 1*, that captures the *name* and *age* of a *Person* sending/printing the output on the computer screen. This program is designed to declare a *structure* (a record), feed data, and output/read data from a record to the screen.

Figure 3 shows how *Program 1* is visualized statically using the MTL RAM diagram. As shown in Figure 3, the execution of instructions line numbers 4, 5, 6, and 7 deals with the structure definition. The definition instantiates the *struct* called *Person* with two-member- fields, *name* of type string, and *age* of type *int*. After defining the structure, the memory status will remain free. Line number 8 begins the execution in the *main function*. Line number 9, *Person Tanzanian,* instantiates a structure labeled *Tanzanian,* reserving space in the RAM to hold a *name* (string) and *age* (integer) concerning any *Person*. At line number 10, data (Salum) is entered in the field *name*, and 28 is entered in the field *age*.

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4  struct Person {//program definition
5  string name;
6  int age;
7  };//the end of program definition

8  int main() {
9  Person Tanzanian; //declaration
10 Tanzanian.name = "Salumu"; //data feeding
11 Tanzanian.age = 28;
12 return 0;
13 }
```

Fig. 2. Program #1

1	#include <iostream>	Proram #1 statement	RAM Status
2	#include <cstring>		Free
3	using namespace std;		
4	struct Person {//program definition		
5	string name;	string name;	free
6	int age;	int age;	free
7	};//the end of program definition	//the end of program definition	free
8	int main() {		reserved
9	Person Tanzanian; //declaration	Person Tanzanian;	reserved
10	Tanzanian.name = "Salumu"; //data feeding	Tanzanian.name	Salum
11	Tanzanian.age = 28;	Tanzanian.age	28
12	return 0;		
13	}		

Fig. 3. Visualization of Program #1 by using MTL visual format

The Machine-driven Component

The machine-driven component consists of a CeliotM, compiler, and tutorial module. CeliotM enables a learner to write a program; compile it; visualize and debug the program that consists of pointers, structures, arrays, linked lists, queues, and stacks. CeliotM further incorporates sort and search modules. As shown in Figure 4, CeliotM consists of four main areas: – (i) *Method Area*, (ii) *Expression Evaluation Area*, (iii) *Constant Area*, and (iv) Instance and *Array Area*. These subdivisions are separated by white-dashed lines appearing as margins between them. Each subdivision has a heading on top of it, describing what type of actors it is displaying. The *Method Area* displays the functions as these are called by the program. The *Expression Evaluation* module shows all actors animating the program expressions and their evaluations. The *Constant Area* generates the constant values in the programs. Finally, the *Instance and Array* module enables the user to see the *Actors* of arrays and data structures instances.

After manual visualization, *Program #1* can be visualized and compiled in CeliotM. CeliotM compiles both linear and nonlinear structures.

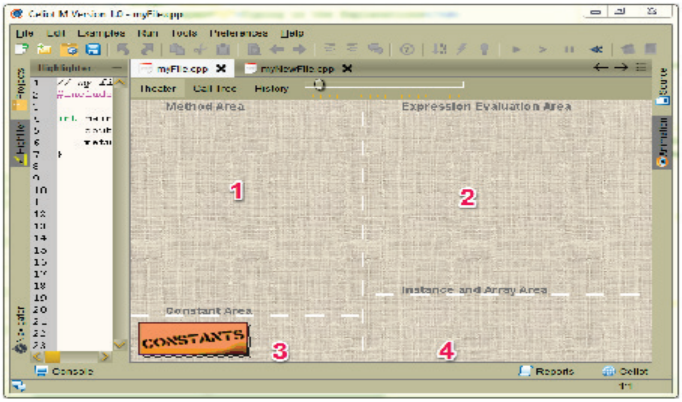

Fig. 4. The Theatre, with its four parts

Figure 5 shows how *Program #1* in Figure 2 is compiled and visualized in CeliotM. In addition, CeliotM shows the memory status after the execution of lines 9 and 10, respectively. This visualization is congruent with its learner-driven version depicted in Figure 3.

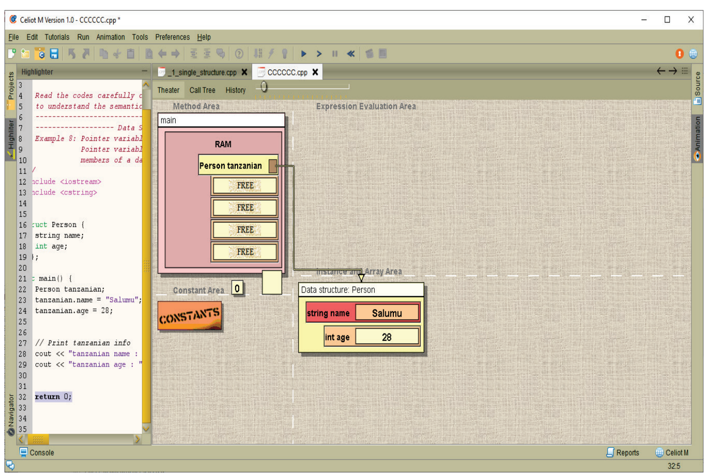

Fig. 5. Visualization of the Program 1 in CeliotM

In using a CeliotM, a learner can effectively engage with the tool. He/She can control the speed of the animation, receive an informative error message, view the dynamic behavior of the program, input new values, and compile and debug the program. While the learner-driven component involves the learner mentally and enhances critical thinking, the machine-driven component provides learners with the actual working and coding space for writing, debugging, compiling, and visualizing programs.

Methodology

Research Approach

This study aimed to evaluate the effectiveness of using the CV framework in learning the Data Structures course. The study employed a mixed-research design approach, combining experiment, documentary review, and survey questionnaire. The experiment evaluated the effectiveness of using the CV framework based on students' test performance scores. The grade analysis method [19][34] was used to evaluate the effectiveness of using the CV framework on students' pass rate (exams final grade). Finally, the survey questionnaire method was used to evaluate students' perceptions on the effectiveness of using the CV framework in learning the Data Structures course.

Study Location and Sample Size

The study was conducted at the College of Informatics and Virtual Education (CIVE) of The University of Dodoma in Tanzania. The experimental sample was 887 first-year students from CIVE enrolled to study CS 122 (data structures) course in the 2018/2019 academic year. These students were pursuing an undergraduate degree in computer science and other ICT-related programs. They all had studied an introductory programming course (CS 110) in C++ in the first semester. Such students, therefore, had basic skills in programming. The examination results of first-year undergraduate students for the academic year 2017/2018 and 2018/2019, respectively, formed the samples for the control and experimental groups. The examination results for the academic year 2017/2018 belong to students who were taught the CS122 course using the traditional lecture method, while the 2018/2019 academic year examination results belong to students who were instructed CS 122 course using the CV framework. Out of 887 students who participated in this study, only 853 completely filled the questionnaire. They were thus considered to be the valid sample for the survey questionnaire.

The Experiment

This study used a single factor within the experiment design, with pre-test and post-test measures based on the test score. The subjects were 887 first-year students studying Data Structures course in the 2018/2019 academic year. The same sample size was used as a control and experimental group, where students' pre-test and post-test performance scores were used as dependent variables. The independent variables were (i) the traditional lecture method and (ii) the CV framework.

Hypotheses

Null hypothesis:

(i) The mean test performance between the traditional lecture method and a combination of the CV framework and lecture method is equal (Hp0): $\mu_{\text{lecture method}} = \mu_{\text{CV framework+leture method}}$. This implies no statistical difference in the students' mean data structures performance between the two approaches.

Alternative hypothesis

(ii) The mean test performance between the traditional lecture method and a combination of the CV framework and lecture method is not equal (Hp1):

$\mu_{\text{lecture method}} \neq \mu_{\text{CV framework+lecture method.}}$ This implies a significant statistical difference in the students' mean data structures performance between the two approaches.

Materials and Tools

The materials and tools used included the data structures syllabus; CeliotM; Borland C++ compiler; learner-driven instructional materials notes, pre-test examination, post-test examination, end-of-semester examination reports, and survey questionnaires.

Procedure

The experiment took place at CIVE between April and July 2019. Each week comprised 8 hours (4-hour lectures, 2 hours of tutorial, and 2 hours of laboratory work. During the experiment, the students were first taught topics of pointers, records, linked lists, recursions, and bubble sort algorithms for three (3) weeks by using the traditional lecture method. Such students were given a set of exercises for review. The students had access to the lab, where they were doing exercises and discussing together in the lab. They were using Borland C++ compilers.

Since all first year, undergraduate students were involved in the experiment, and lecture sessions were done during the weekend, the students were allowed to practice 4 hours per week; i.e., one hour per week was used for lab time under the instructor's supervision; one hour in the laboratory for self-study; and two hours using their computers during class time. Borland C++ compiler was installed in all computers in all laboratories at CIVE. The lab was also available for students to practice what they were learning at their own pace. To ensure that the students were doing laboratory work, the researcher participated in the laboratory sessions. Students were also filling in an activity log to record the exercises they did in the lab. During the laboratory sessions, the students were required to do exercises and review examples using the traditional approach with a Borland C++ compiler.

After learning these topics for three (3) weeks, students were given a class assignment as a pre-test to judge their understanding before learning data structures using the CV framework. Then, the researcher trained all participants on how to use the CV framework and provided learner-driven instructional materials on the selected topics of pointers, records, linked lists, recursion, and bubble sort algorithms. CeliotM was installed on all computers in all five laboratories, including the libraries at CIVE.

The students used the CV framework for another three (3) weeks to practice for the given data structures topics. They sat for a post-test to verify if the use of the CV framework provided better knowledge gain than the traditional lecture method in the selected topics. Both pre-test and post-test were marked out of 30. The test duration was 60 minutes. Out of 887 students who participated in the experiment, 879 students did both pre-test and post-test. After doing the pre-test and post-test, the mean performance data obtained before and after using the CV framework were recorded for analysis. Figure 6 shows the pre-test and post-test arrangement.

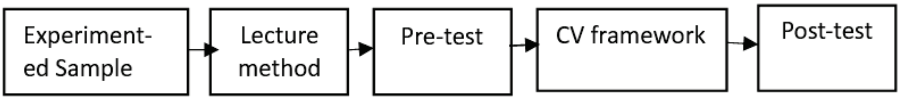

Fig. 6. Pre-test and Post-test Arrangement

The students continued using the CV framework for eight more weeks covering insertion sort, merge sort, quick sort algorithms, queue, and stack. Later, they did the end-of-semester examination. After doing the end-of-semester examinations, the data pertaining to final examination performance were collected for analysis.

Data Collection Methods

In this study, a questionnaire with closed-ended questions was used to collect data on students' perceptions of the effectiveness of the CV framework in teaching and learning data structures. The questionnaire was self-administered and was given physically to respondents and collected within twenty minutes after being filled in. Data from the experiment were collected by using pre-test and post-test. Students' examination reports for the academic years 2017/2018 and 2018/2019 were obtained from CIVE examination office after getting a data collection permit from the University of Dodoma. Data pertaining to the student's examination pass rates (final examination grades) was extracted from examination reports for analysis.

Data Analysis

Pre-test and post-test data were statistically analyzed using the paired t-test. Data from examination reports were analyzed by using the grading analysis method. That is, the data pertaining to the 2017/2018 academic year students' final examination grades (control group) was analyzed by comparing it with the data of 2018/2019 academic year students' final examination grades (the experimental group). This method has been adapted from [19, 34]. The results of the comparison were conveyed using descriptive statistics (percentages). Quantitative data from the questionnaire were analyzed by calculating the average score per question based on a 5-point Likert scale. Likert scale scores were calculated based on the average score per question. Questions were assigned responses from 0 to 4, with 0 representing "strongly disagree," 1 representing "disagree," 2 representing "undecided," 3 representing "agree," and 4 representing "strongly agree." "Final scores were then normalized and calculated as an average and then normalized as a percentage of the mean maximum score. The closed-ended questions, which were later coded and analyzed, were expressed as percentages using descriptive statistics.

It was predicted that using the CV framework in learning data structures would improve students' test performance compared to the traditional approach. It was also expected that using the CV frameork would improve students' pass rate (final examination grades) and their perfection in learning data structures and thus reduce failure rate compared to the previous year when only the traditional lecture method was used.

Results

This section presents the results of the study from the experiment, document analysis, and questionnaire survey.

Results from the experiment

Table 1 shows the group statistics between the post-test and pre-test results. Table 2 summarizes the significance levels between the pre-test and post-test (i.e., the p-value is) 0.000 with t=38.325, with a mean difference of 7.2179 in students' classwork scores. Since the p-value is below the significance level (i.e., 0.05), the difference between the two means is statistically significant. Therefore, the alternative hypothesis is accepted, i.e., a significant statistical difference was found between the two means of classwork performance. These results indicate that the use of the CV framework had an impact on data structures cognition.

Table 1

Group Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	POST	18.761	879	6.2372	0.2104
	PRE	11.543	879	5.0226	0.1694

Table2

Paired Sample Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	POST – PRE	7.2179	5.5838	0.1883	6.8482	7.5875	38.325	878	0.000

Results from the Documentary Analysis of the Students' Final Examination Grades

Table 3 presents the comparison of students' pass rate and final examination grades at CIVE for the 2017/2018 academic year (when only the traditional lecture method was used) and the 2018/2019 academic year (when a combination of both traditional lecture methods and CV framework was used).

Table 3

Comparison of CS 122 Grades between the 2017/2018 and 2018/2019 academic years at CIVE

	YEAR	TOTAL	A	A, %	B+	B+, %	B	B, %	C	C, %	D	D, %	E	E, %
CIVE	2017/2018	826	1	0.1	43	5.2	206	24.9	346	41.9	115	13.9	115	13.9
	2018/2019	887	19	2.1	114	12.9	254	28.6	332	37.4	80	9.0	88	9.9

Table 3 shows that the percentage of A, B+, and B grades were largely improved after using the CV framework in the academic year 2018/19. The total number of A's for students who did the final examination increased from 0.1% in 2017/18 to 2.1% in the academic year 2018/19, while that of B+`s increased from 5.2% to 12.9%. Furthermore, the pass rate also increased from 71.1% to 81.2% in the academic year 2018/19. While the students' failure rate was 27.8% in the academic year 2017/18, there was a failure rate of 18% in the academic year 2018/19, which implies that after using the CV framework, the failure rate in the CS 122 (data structures) course at CIVE was reduced by 9.0%. These results suggest that the use of the CV framework improved data structures comprehension among the students; and hence the overall pass rate.

Students' perceptions towards learning data structures after Using the CV Framework

Table 4 summarizes the results of students' perceptions of learning data structures after using the CV framework. As shown in Table 4, the students' average responses for each question was above 70%, implying that students' responses from the questionnaires concerning students' perceptions after using the CeliotM framework towards learning the CS122 course have largely improved

Table 4

Students' perceptions towards Learning CS2 after Using the CV Approach

Question statement	Average Score, %	No of respondents
Using the CV approach motivated me to learn the CS122 course	75.9	852
Using the CV approach helped me to learn the CS122 course without rote learning	71.2	852
Using the CV approach improved my confidence in learning the CS122 course	75.8	853
Using the CV approach helped me learn the CS122 course with less mental effort	72.4	853
I found learning the CS122 course easier after using the CV approach	70.4	852

Discussion

Findings from the study have shown that using the CV framework among the students who were studying a CS122 Data Structures course at CIVE has helped improve the pass rate up to 81.2% while improving upper grades by 10.44% compared to the traditional approaches. Findings from the study have further revealed that the use of the CV framework assisted students in improving their CS122 comprehension, confidence, and motivation to learn CS122. It also helped them learn CS 122 with less anxiety and less mental effort than the traditional lecture method. One possible reason that the CV framework improved students' academic performance is that the CV framework provides the maximum level of learner engagement. This is because the CV framework requires learners to manually write algorithm steps, apply pseudocode, visualize programming threshold concepts based on the MTL approach, and then use CeliotM for visualization and compilation. This dual working style helped learners understand the inner workings (programming logic) of programming constructs in terms of algorithm steps and thus helped them easily convert algorithms into codes when writing computer programs. These results suggest that the dual use of a congruent pair of learner-driven and machine-driven instructional materials improves student success rates while decreasing failure rates, as shown in these research findings. The results of this study confirm [35], multimedia learning theory [36], and [37], that the dual use of visual and nonverbal channels enhances learning.

Conclusion

This study evaluated and confirmed the effectiveness of the CV framework in data structures teaching and learning. The CV framework is both manual and machine-driven. Machine-driven systems combine AV, PV, and compilation. Learning-driven systems use MTL RAM diagrams to visualize the dynamic properties of the code. Results show that using the CV framework increases motivation and reduces CL, leading to better performance. The results suggest that PV tools can be better used in learning data structures if they are integrated with common compilers and compatible with learner-driven teaching materials. However, the evaluation of the effectiveness of the proposed approach was conducted within a short period of time. Therefore, further longitudinal studies with real experiments are needed to validate further the effectiveness of the CV framework in teaching and learning data structures.

References

1. Gárcia-Mateos G., Fernández-Alemán J.L. A course on algorithms and data structures using on-line judging. *Proc Conf Integr Technol into Comput Sci Educ ITiCSE*. 2009;45–9.
2. Pérez-sánchez B., Morais P. *Learning Data Structures – Same Difficulties in Different Countries*? 2016;8540(c).
3. Qian Y., Lehman J. Students' Misconceptions and Other Difficulties in Introductory Programming : *A Literature Review*. 2017;18(1):1–24.
4. Krause-Levy S., Valstar S, Porter L., Griswold W.G. A demographic analysis on prerequisite preparation in an advanced data structures course. *ACM Inroads*. 2022;13(2):34–41.
5. Gárcia-Mateos G., Fernández-Alemán J.L. A course on algorithms and data structures using on-line judging. In: *Proceedings of the 14th annual ACM SIGCSE conference on innovation and technology in computer science education*. 2009:45–9
6. Fouh E., Akbar M., Shaffer C.A., Tech V. *The Role of Visualization in Computer Science Education*. 2012.
7. Naps T.L., oßling G., Almstrum V., Dann W., Fleischer R., Hundhausen C., et al. Exploring the Role of Visualization and Engagement in Computer Science Education Report of the Working Group on &quot. *Improving the Educational Impact of Algorithm Visualization " Acm*. 2002;35(2):131–52.

8. Sobh T. *A tool for data structure visualization and user-defined algorithm animation.* 2014;(October 2001).
9. Pieter W. How to optimize cognitive load for learning from animated modelse. *The Netherlands Organisation for Scientific Research*. 2007.
10. Romanowska K., Singh G., Dewan M.A.A., Lin F. Towards Developing an Effective Algorithm Visualization Tool for Online Learning. 2018 IEEE *SmartWorld, Ubiquitous Intell Comput Adv Trust Comput Scalable Comput Commun Cloud Big Data Comput Internet People Smart City Innov*. 2018;(October):2011–6.
11. Naps T., Guido R., Darmstadt T.U., College M., Dann W., Cooper S., et al. *Evaluating the Educational Impact of Visualization*. 2003
12. Sorva J., Karavirta V., Malmi L. *A Review of Generic Program Visualization Systems for Introductory*. 2013;13(4).
13. Vagianou E. Program Working Storage: *A Beginner's Model*. 2006;69–76.
14. Bellström P., Thorén C. Learning how to program through visualization: A pilot study on the bubble sort algorithm. *2nd Int Conf Appl Digit Inf Web Technol ICADIWT* 2009. 2009;90-4.
15. Myller N., Bednarik R., Sutinen E. Extending the Engagement Taxonomy: *Software Visualization and Collaborative Learning*. 2009;9(1).
16. Jonathan F.C., Karnalim O., Ayub M. *Extending The Effectiveness of Algorithm Visualization with Performance Comparison through Evaluation-integrated Development*. 2016;16–21.
17. Petre M., de Quincey E. A gentle overview of software visualisation. *PPIG News Lett*. 2006;1–10.
18. Price B.A., Baecker R.M., Small I.S. A principled taxonomy of software visualization. *J Vis Lang Comput*. 1993;4(3):211–66.
19. Scott A.S. *Using Flowcharts, Code and Animation for Improved Comprehension and Ability in Novice Programming Certificate of Research*. 2010;(March).
20. Brown M.H., Sedgewick R. Techniques for algorithm animation. *Ieee Softw*. 1985;2(1):28.
21. Maloney J., Resnick M., Rusk N., Silverman B., Eastmond E. The scratch programming language and environment. *ACM Trans Comput Educ*. 2010;10(4):1–15.
22. Moreno A., Myller N., Sutinen E., Ben-Ari M. Visualizing programs with Jeliot 3. In: *Proceedings of the working conference on Advanced visual interfaces*. 2004:373–6.
23. Bergin J., Patiho-matt M., Brodlie K., Mcnally M., College A., Goldweber M., et al. An overview of visualization: *Report of the Working Group on Visualization*. 1996;192–200.
24. Hidalgo-Céspedes J., Marín-Raventós G., Lara-Villagrán V. Learning principles in program visualizations: *A systematic literature review. Proc – Front Educ Conf FIE*. 2016;2016-Novem.
25. Mselle L. Using Memory Transfer Language (MTL) as a Tool for Program Dry-running. *Int J Comput Appl*. 2014;85(9):45–51.
26. Hidalgo-Céspedes J., Mar\'\in-Raventós G., Lara-Villagrán V. Learning principles in program visualizations: *A systematic literature review. Proc – Front Educ Conf FIE*. 2016;2016-Novem.
27. Sorva J., Karavirta V., Malmi L. A review of generic program visualization systems for introductory programming education. *ACM Trans Comput Educ*. 2013;13(4).

28. Colaso V., Kamal A., Saraiya P., North C., Mccrickard S., Shaffer C.A. Learning and Retention in Data Structures : *A Comparison of Visualization, Text, and Combined Methods*. 2005;1–2.
29. Osman W.I., Elmusharaf M.M. Effectiveness of Combining Algorithm and Program Animation : *A Case Study with Data Structure Course*. 2014;11:155–68.
30. Nathasya R.A., Karnalim O., Ayub M. Integrating program and algorithm visualisation for learning data structure implementation. *Egypt Informatics J* [Internet]. 2019;(November). Available from: https://doi.org/10.1016/j.eij.2019.05.001
31. Pathania U., Singh A. International Journal of Software and Web Sciences (IJSWS) *Visualization Tool for Tree and Graph Algorithms with Audio Comments*. 2014;51–8.
32. Silva D.B., Aguiar R.D.L., Deconto D.S., Jr C.N.S. *Recent studies about teaching algorithms (CS1) and data structures (CS2) for computer science students*. 2019.
33. Burkhard R.A. Towards a framework and a model for knowledge visualization: Synergies between information and knowledge visualization. *Knowl Inf Vis Search Synerg*. 2005;238–55.
34. Iqbal Malik S., Coldwell-Neilson J. Impact of a New Teaching and Learning Approach in an Introductory Programming Course. *J Educ Comput Res*. 2017;55(6):789–819.
35. Clark J.M., Paivio A. Dual coding theory and education. *Educ Psychol Rev*. 1991;3(3):149–210.
36. Mayer R., Mayer R.E. *The Cambridge handbook of multimedia learning.* Cambridge university press; 2005.
37. Shaffer D. Applying Cognitive Load Theory to Computer Science Education. 2014;(January 2003).

Information about the author:

Adam B. Mtaho, PhD, Lecturer, Department of in Information and Communication Technology, Arusha Technical College, P. O. Box 296, Arusha, Tanzania.
E-mail: abasigie@.yahoo.com

Leonard J. Mselle, PhD, Associate Professor, Department of Computer Science and Engineering, The University of Dodoma (Dodoma, P. O. Box 296, 255, Arusha, Tanzania).
Email: mselel@yahoo.com

Masoud M. Masoud, PhD, Lecture, Department Computer Studies (Dar es Salaam Institute of Technology, P. O. Box 2958, Dar es Salaam, Tanzania).
Email: bigutu@gmail.com

ВЛИЯНИЕ ИСПОЛЬЗОВАНИЯ КОНГРУЭНТНОГО ПОДХОДА К ВИЗУАЛИЗАЦИИ НА ИЗУЧЕНИЕ КУРСА CS2

***Адам Басиги Мтахо*[1], *Леонард Дж. Мселле*[2], *Масуд М. Масуд*[3]**

[1] *Технический колледж Аруши, Танзания, abasigie@.yahoo.com*

[2] *Университет Додомы, Додома, Танзания, mselel@yahoo.com*

[3] *Дар-эс-Саламский технологический институт, Дар-эс-Салам, Танзания, bi-gutu@gmail.com*

Аннотация. Большинство преподавателей информатики (И) согласны с тем, что изучение курса структур данных и алгоритмов (CS2) является сложным процессом из-за его высокой когнитивной нагрузки. Следовательно, изучение CS2 было связано с высоким уровнем неуспеваемости. Чтобы помочь снизить высокую когнитивную нагрузку, возлагаемую на учащихся изучающих CS2, были предложены алгоритмы визуализации (АВ). Несмотря на длительное использование АВ в преподавании и изучении CS2, исследования показывают, что такие инструменты не были столь педагогически эффективными, как это ожидалось. Это исследование было направлено на изучение влияния использования подхода конгруэнтной визуализации (КВ) в преподавании и изучении CS2. В этом подходе используется комбинация двух совместимых инструментов визуализации программ, которые включают подходы, управляемые машиной, и подходы, управляемые учащимся. Воздействие использования КВ-подхода оценивалось с использованием комбинации экспериментов, анализа документов и методов анкетирования. Объектами исследования стали 887 студентов первого курса бакалавриата Колледжа информатики и виртуального образования (CIVE) Университета Додомы в Танзании, изучающих CS2. Результаты показывают, что использование подхода КВ улучшило как показатели непрерывной оценки учащихся, так и показатели результатов сдачи выпускных экзаменов по сравнению с традиционным подходом. Ответы учащихся на последующий опрос показали, что использование подхода CV повысило мотивацию и уверенность учащихся в обучении CS2.

Ключевые слова: *структура данных и алгоритм, курс CS2, визуализация программы, визуализация алгоритма, конгруэнтная визуализация*

Для цитирования: Mtaho A.B., Mselle L.J., Masoud M.M. The Effectiveness of Using a Congruent Visualization Framework on Learning a Data Structures Course // Education & Pedagogy Journal. 2023. Вып. 1 (5). P. 60–76. doi: 10.23951/2782-2575-2023-2-60-76

Информация об авторе:

Адам Б. Мтахо, доктор философии, преподаватель кафедры информационных и коммуникационных технологий, Арушский технический колледж (почтовый ящик

296, Аруша, Танзания).
E-mail: abasigie@.yahoo.com

Леонард Дж. Мселле, доктор философии, доцент кафедры компьютерных наук и инженерии, Университет Додомы (Додома, почтовый ящик 296, 255, Аруша, Танзания).
E-mail: mselel@yahoo.com

Масуд М. Масуд, доктор философии, лекция, кафедра компьютерных исследований, Дар-эс-Саламский технологический институт (почтовый ящик 2958, Дар-эс-Салам, Танзания).
E-mail: bi-gutu@gmail.com

Submitted April 1, 2022