I.J. Modern Education and Computer Science, **2025, 6, 1-17**
Published Online on December 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2025.06.01

Eliciting Knowledge Transfer and Self-management Skill through the Effects of Cognitive Load Theory on Programming Learning

Carlos Sandoval-Medina*
Department of Information Systems, Autonomous University of Aguascalientes, Aguascalientes, 20100, México
E-mail: al285667@edu.uaa.mx
ORCID iD: https://orcid.org/0000-0002-6552-4150
*Corresponding Author

Estela L. Muñoz-Andrade
Department of Electronic Systems, Autonomous University of Aguascalientes, Aguascalientes, 20100, México
E-mail: lizbeth.munoz@edu.uaa.mx
ORCID iD: https://orcid.org/0000-0003-4182-5044

Carlos A. Arévalo-Mercado
Department of Information Systems, Autonomous University of Aguascalientes, Aguascalientes, 20100, México
E-mail: carlos.arevalo@edu.uaa.mx
ORCID iD: https://orcid.org/0000-0002-8349-7985

Jaime Muñoz-Arteaga
Department of Information Systems, Autonomous University of Aguascalientes, Aguascalientes, 20100, México
E-mail: jaime.munoz@edu.uaa.mx
ORCID iD: https://orcid.org/0000-0002-3635-7592

Received: 01 November, 2024; Revised: 20 March, 2025; Accepted: 26 August, 2025; Published: 08 December, 2025

Abstract: Cognitive Load Theory (CLT) is an instructional design theory that aligns with human cognitive architecture for creating instructional materials, through the design guidelines of its 17 instructional effects. However, the Self-Management effect suggests that students can be instructed to manage their learning. The Collective Working Memory effect highlights how a group of students working together can foster a more effective learning environment than an individual student, resulting in better learning outcomes. This research explored applying the Self-Management effect of CLT alongside the Collective Working Memory effect learning data structures in basic programming and measuring their effectiveness regarding essential knowledge acquisition in declarative knowledge, knowledge transfer (near transfer) in procedural knowledge, and developing self-management skills. Cognitive load was measured to determine the difference between groups and to determine the correlation with learning outcomes. The study was carried out through a quasi-experimental design with homogeneous groups, involving students from the Autonomous University of Aguascalientes. The results suggest positive findings in knowledge transfer as well as the development of self-management skills. The cognitive load between the participating groups does not show any significant statistical difference, nor does it show any correlation with the learning results.

Index Terms: Cognitive Load Theory, Self-Management Effect, Collective Working Memory Effect, Computer Education, Self-Management Skill, Knowledge Transfer

1. Introduction

A key process in education is knowledge transfer (KT), whereby information flows from knowledge providers (teachers, books, video tutorials, online courses) to knowledge recipients (students), who then apply that knowledge to solve problems in the real world [1]. Knowledge transfer in programming refers to the capacity to apply learned

abilities and information to novel contexts. The literature identifies two concepts in knowledge transfer: near transfer, which happens in similar situations, like applying code from a workshop exercise to a real-world task, and far transfer, which happens in entirely different situations, like when engineers and scientists use computational methods to tackle new disciplinary problems [2,3]. For the purposes of this study, knowledge transfer refers to near transfer. However, programming is a difficult cognitive task for beginners since it calls for both declarative knowledge (conceptual understanding) and procedural knowledge (procedural abilities) [2].

For this reason, in a quality learning process, the instructional material accessed by students should be based (among other elements) on evidence-based instructional design practices and theories [5]. However, due to the vast amount of information available on the internet, students often encounter resources that lack a theoretical foundation [5,6]. Therefore, students should understand the underlying conceptual principles that comprise good quality learning materials and be able to adapt them to their workflow and manage their learning effectively [6].

Cognitive Load Theory (CLT) is an instructional design theory based on human cognitive architecture focused on supporting teachers to optimize the design of instructional materials, creating, modifying, or adapting materials that reduce mental effort so that students learn efficiently [7]. In the context of CLT, the Self-Management effect refers to students being instructed to use strategies, such as CLT principles, to manage their working memory load in the learning process [8]. Applying CLT principles themselves to poorly designed instructional material to optimize their learning [5].

The Collective Working Memory effect refers to the idea that a group of collaborating students can be viewed as one information processing system. This can create a more effective collective environment than that of a single student, where not all members of the group need to process all the information simultaneously, thus complementing each other's knowledge [9].

2. Literature Review

2.1. Cognitive load theory (CLT)

CLT is an instructional design theory [5,10,11] based on evolutionary [12], human cognitive architecture [13] and Bartlett's schema theory [14]. In its simplest form, this theory identifies that information processing in the human brain goes through two subsystems: working memory, which is very limited, and long-term memory, which is unlimited.

According to CLT, any learning process requires that information from the environment must first be processed in working memory, which has limited capacity. That is, a person's working memory can only hold a small number of discrete elements at a time and for a brief period. Through this interchange between memory subsystems, schemas are constructed in long-term memory, which in turn has virtually unlimited capacity. These schemas can endure for a lifetime and may eventually become automated through constant and conscious practice and requiring minimal memory resources [5,15]. The cognitive load refers to the number of resources that working memory requires for a learning process [16].

The traditional CLT framework, referred to as "old CLT" in publications from 1998 to 2010 [10,15], conceptualized cognitive load as comprising three distinct and additive components:

- Intrinsic Load (IL). This reflects the cognitive demands that come with the learning material's intrinsic complexity and element involvement. For a given learning task and learner combination, IL is essentially unchangeable since it is based on the learner's past knowledge and the nature of the content.
- Extraneous Load (EL). This includes the mental strain caused by poor presentation and instructional design. Better instructional design can lessen EL, which is regarded as "wasteful" cognitive strain that does not aid in learning.
- Germane Load (GL), which is the load necessary for learning–this is “good” cognitive load, representing the resources that working memory dedicates to the learning process [15]. The above statement was modified in later publications from 2010 onwards, called new CLT, where the germane load is discarded as a third load, therefore, the cognitive load is only extraneous or intrinsic [5,15].

The contemporary CLT framework or new CLT from 2010 to present, represents a significant paradigm shift that fundamentally altered our understanding of cognitive load mechanisms [5,15,16]. In new CLT, Germane load has been reconceived as the efficient use of working memory resources to manage item interaction, which is one of the major theoretical shifts in Cognitive Load Theory [5,15,16]. The working memory resources available to process item interactivity after considering intrinsic and extraneous demands are now recognized as germane cognitive load, while the new CLT makes it clear that only intrinsic and extraneous loads are truly additive, calculating total cognitive load as the sum of these two components [16]. A more nuanced understanding of load interactions is established by the revised theory, which shows that while reducing intrinsic load through improvements in instructional design increases available germane cognitive resources [5,15,16], an increase in extraneous load directly reduces the working memory resources to handle intrinsic load [15], which in turn decreases germane cognitive processing capacity [16].

The new CLT's theoretical implications for educational practice include increased design precision and clearer guidance for instructional designers by concentrating on two primary objectives: managing intrinsic cognitive load through appropriate task selection and sequencing [16,17] and minimizing extraneous cognitive load through improved instructional design [5,17], as opposed to trying to simultaneously optimize three competing types of loads. By offering more accurate operational definitions for cognitive load assessment, the reconceptualization overcomes earlier measurement issues where researchers found it difficult to objectively differentiate between intrinsic and germane pressure [5,15]. Additionally, educators can make better decisions about when and how to introduce complex learning tasks by viewing germane load as available working memory capacity rather than additional load [16,17]. This is especially important for programming education, where item interactivity is naturally high [15,16].

According to research demonstrating that poorly designed programming materials can impose significant extraneous cognitive load [15,16], the new CLT framework's implications for programming education suggest that instructional materials should primarily focus on eliminating extraneous cognitive load (through clear presentation, integrated examples, and reduced redundancy) rather than attempting to directly manipulate germane load [5,15,17].

Since students immediately increase their germane cognitive processing capacity without adding cognitive load when they learn to recognize and reduce superfluous load in their study materials, the Self-Management impact optimization technique fits in especially well with the new CLT principles [5,6,18]. Additionally, distributed cognitive processing among group members effectively increases the total available working memory capacity to handle item interactivity [5,9,17], which is especially advantageous in programming contexts where complex problem solving requires extensive information processing [9]. This is why the new CLT theoretically supports the Collective Working Memory effect.

Both versions of CLT are still used in research. In computer education, old CLT is more common. In this research, the new CLT is applied due to its theoretical perspective, where total cognitive load is determined by total element interactivity generated by intrinsic and extraneous cognitive load, and applying guidelines to optimal material design can reduce extraneous cognitive load, freeing up resources in working memory to increase germane cognitive load [17].

Either way, the focus of the CLT remains: to apply instructional principles to reduce working memory load imposed by irrelevant elements on instructional materials that hinder learning, thus freeing up working memory resources for activities that enhance learning [18].

2.2. Self-Management effect

Self-Management effect suggests that students can be instructed to actively employ strategies to manage the load on their working memory during learning [7] by utilizing these strategies through the principles of CLT. This effect stems from the universal access of knowledge, where students not only have access to materials designed by the instructor but also to the vast repository that is the internet [5]. However, many of these resources are not optimally designed [7], so students must learn how to adapt or improve these materials by themselves for better learning performance [6].

The focus of Self-Management effect, unlike other instructional effects, is to provide students with strategies to manage their self-learning skills when they face instructional materials with redundancy, split-attention, transient information, and other characteristics that induce extraneous cognitive load [17]. For example, some recommended strategies would be connecting separate information (split-attention) through notes, drawing arrows, highlighting information, integrating text and images, or eliminating repetitive information (redundancy) [18].

In recent literature, the Self-Management effect has mostly been studied in combination with the split-attention effect [3]. Other studies have demonstrated the benefits of self-management strategies for learning in different areas such as mathematics, science, and accounting [7,18,19,20,21,22], all of them in combination with only the split-attention effect, but these reports suggest that the Self-Management effect can also be combined with other effects [7].

2.3. Collective Working Memory effect

The Collective Working Memory effect describes how a group of students working collaboratively functions as a unified information processing system, combining multiple working memories and varying levels of prior knowledge. This collective approach can create a more effective learning space than individual study and leads to improved learning outcomes [5]. Since group members don't need to process all information simultaneously [9], students who struggle with specific topics can receive peer support. This creates interdependence among group members in both generating and acquiring knowledge [5,9,17].

The intrinsic cognitive load generated by complex tasks or information in knowledge transfer can be distributed among collaborative learning group members, effectively increasing their collective cognitive capacity. However, this collaboration requires communication and coordination between participants, which demands additional cognitive effort [5]. Nevertheless, when groups achieve an optimal balance between distributed information processing and the mental effort required for transactive activities, the Collective Working Memory effect emerges [9].

In contrast to individual learning, collective working memory allows members of a group to share the load of processing information, potentially increasing the total cognitive capacity available, particularly for complicated tasks. Contrarily, transaction costs are the strain placed on each person's working memory capacity because of the coordination and communication needed for group tasks. A balance between the expenses associated with transactional tasks and the sharing advantage—the advantages of collective working memory—is necessary for effectiveness. It can

be advantageous for students who have little prior knowledge to learn new information from their peers, particularly in diverse or incompletely informed groups [9].

However, because teamwork and cooperation require coordination and communication, they place an additional strain on each person's working memory capacity. Groups may exhibit learning-detrimental behaviors such convoluted conversations, trouble reaching consensus, social loafing, interpersonal disputes, and sluggish decision-making. This is why it's crucial to create diverse teams and have the teacher provide ongoing support as the process progresses [9].

2.4. Some instructional effects of CLT related to the study

As the theory developed, several "effects" were reported by the authors and researchers. As such, 17 effects have been identified that can be applied to different instructional conditions, which aim to reduce extraneous cognitive load in learning materials or to modify the intrinsic cognitive load of learning tasks [5], with the majority reporting positive empirical results [16]. Among these effects, the most cited effects of the CLT are the worked example and the completion problem [23,24,25]. Table 1 shows some effects of CLT related to this document. However, it is important to note that CLT effects do not work in isolation. For example, the element interactivity effect acts as a unifier by influencing the total cognitive load (intrinsic + extraneous). Other effects, such as the worked example, variability, and guidance fading effects, can be enhanced or diminished depending on how element interactivity affects the total cognitive load.

Table 1. Some instructional effects of CLT, developed from [5,10,15,16,26,27].

Instructional Effect	Description
Split-Attention Effect	The split-attention effect arises when two or more critical sources of information are either physically or temporally separated during presentation to the learner, increasing cognitive load and hindering knowledge transfer. To alleviate the split-attention effect, it is crucial to present related sources of information in an integrated format.
Redundancy Effect	The redundancy effect occurs due to split-attention and the modality effect when the same information is presented to the student in different formats, such as text and audio simultaneously. Therefore, it is crucial to identify redundant independent sources of information and assess them to select the most effective one for application, thereby eliminating redundancy.
Transient Information Effect	Transient information refers to data that vanishes before students can fully process it. Spoken text, instructional videos, or animations are transient, disappearing after just a few seconds during the teaching process. Long, complex information should not be presented in a transient format.
Element Interactivity Effect	It happens when cognitive load effects are achievable through high element interactivity information vanish or reverse with low element interactivity material. When designing learning materials, it is important to consider the level of element interactivity and to use instructional strategies that are appropriate for the task.

Numerous applications of Cognitive Load Theory in programming education are demonstrated by the examined papers. Vieira, Magana, Falk, and Garcia [28] showed encouraging outcomes when they used worked examples with self-explanations using "glass box" and "black box" approaches (self-explanation, worked example). Using the PyKinect tool with parsons' problems and self-explanations, Fabic, Mitrovic, and Neshatian [29] discovered notable differences in PRE and POST tests. Yen and Wang [30] created a prototype in C++ that asks for compilation error explanations by comparing them to a database, achieving 84% and 78% accuracy (self-explanation). Zhi, Chi, Barnes, and Price [25] found no significant differences using "The Peer Code Helper" with worked examples and self-explanations in block-based programming. Sands [24] reported positive results combining worked examples, pair programming, and live coding (worked example, modality, guide fading, collective working memory). Price, Williams, Solyst and Marwan [31] found no discernible changes between the two tools, and Winter, Friend, Matthews, Love and Vasireddy [32] reported positive outcomes using the Bricklayer tool, which focuses on minimizing superfluous cognitive burden without mentioning specific consequences.

In addition, Berssanette and de Francisco [16] presents a systematic review of the literature on Cognitive Load Theory (CLT) applied to the field of teaching and learning computer programming, analyzing studies published between 2010 and 2020. The studies relate CLT to programming instruction through cognitive load measurement, resource/tool-based instructional design, diverse pedagogical strategies, specific CLT concepts. Highlighting the relevant or more applied effects, the effect of worked example, which reduces the cognitive load through complete solutions, the effect of completion problem, which are partially worked examples, the effect of split-attention, which seeks to avoid sources of information physically or temporally separated and the effect of modality using independent processors (auditory/visual).

2.5. Knowledge transfer in education and types of knowledge in programming

The idea of knowledge transfer in any teaching-learning process suggests that students learn from various information sources (teachers, books, video tutorials, and online courses), then apply that knowledge to real-world problems to transfer it [1]. Where learning outcomes and knowledge transfer are primarily impacted by two interconnected factors: Teacher Knowledge Transfer (TKT) and Student Absorption Capacity (CA). The ability of individuals to recognize, obtain, and use valuable knowledge is known as CA. The process by which teachers use a variety of instructional modalities to impart to students both explicit (objective and codifiable) and tacit (personal experience difficult to codify) information is known as TKT [1,33,34].

In the context of programming, knowledge transfer refers to the ability to correctly apply acquired skills, knowledge and behaviors to new situations or contexts, hence the concepts of near transfer and far transfer. Near transfer is applied directly when a perceptually similar situation is presented, for example, students are expected to achieve a near transfer of coding skills from a workshop exercise to a practical task, or from a classroom example to a classroom task. Far transfer knowledge must be applied in a context that is different from any previous context where that knowledge was learned and used, e.g. for scientists and engineers, far transfer manifests itself in the application of computational and data science approaches to novel or unfamiliar disciplinary challenges [2,3]. Accordingly, for the purposes of this study, knowledge transfer refers to near transfer.

Programming is more than just learning a language; it's a complex skill that requires both conceptual knowledge of the language and procedural skills to apply it to solving problems. Novices face a unique challenge as they must master two essential categories of interconnected knowledge [2].

Declarative programming knowledge is the "what" of programming. It encompasses knowledge of programming language syntax, constructs, and concepts. It is closely linked to comprehending programs and syntactic facts and semantic meanings. It is essential for developing procedural knowledge through examples like knowing how to use "for" loops or the distinction between i++ and ++i. Procedural programming knowledge, on the other hand, is the "how to" program and refers to the capacity to arrange rules and use algorithms to accomplish specific objectives. It is closely linked to coding abilities and is constructed from the explicit depiction of algorithmic planning and coding procedures, which are improved by observing flowcharting and expert coding procedures [2].

Consequently, it is suggested that the cognitive load theory be used to assist students in improving their Absorptive Capacity, as well as their capacity to recognize and arrange crucial information and to manage their learning (self-management abilities) by employing techniques based on the Self-Management effect. Through the application of the Collective Working Memory effect, they apply knowledge and ideas to produce new ideas, absorb information more effectively, and enhance their comprehension. This enhances knowledge transfer (near transfer). Teachers use a variety of educational modalities to gain from their Teacher Knowledge Transfer.

3. Method

This research combines the Self-Management effect with the Collective Working Memory effect. To apply the Self-Management effect, a self-management guide was developed to teach students how to apply Cognitive Load Theory (CLT) learning strategies through the Self-Management effect to instructional materials that address redundancy, split-attention, and transient information effects. Then, applying the Collective Working Memory effect, students worked in teams following this guide to create their study materials to develop self-management skills. The effectiveness of this approach in learning C++ programming language structures was evaluated through two types of assessments: one focusing on theoretical concepts and another on practical problem-solving applications. Self-management skill development was measured by comparing student-created materials with instructor-developed materials through a rubric that evaluated students' self-management ability, based on guidelines from the self-management guide, where the scores ranged from 0 to 10. While cognitive load was measured using an adapted version of the CS Cognitive Load Component Survey [35].

The research methodology consisted of two main steps. The first step involved developing a self-management guide for programming and preparing treatment materials. The second step comprised implementing the treatment and evaluating results through a five-phase quasi-experimental design.

3.1 Step 1: Design and preparation of material

A programming self-management guide was developed based on the Self-Management effect principles of CLT, addressing redundancy, split-attention, and transient information effects. The guide instructs students on developing their study materials from instructional content that contains design issues related to these effects. The guide consists of three main strategies, which are outlined below:

1. If the material contains only text:
    a. Complement with visualizations such as drawings or concept maps, alluding to the cited text.
    b. Eliminate non-essential or redundant information, generate textual or visual summaries, and indicate essential information by highlighting texts or images. For example, the topic arrays: “An array is a group of memory locations related to each other by the fact that they all have the same name and are of the same type. To refer to a particular position or element within the array, we specify the name of the array and the position number of the element within it". Graphic example of how the elements of a numeric array of size 6 are stored, that is, it stores 6 different numbers (see Fig. 1):

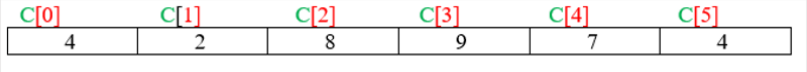

Fig. 1. Graphic example of a numeric array.

Where C is the name of the array and the value between [], the position of each element in the array. For example, the element in C[2] would be 2.

c. In the case of code example texts, it is advisable to accompany the code text with textual or graphic explanations (desktop test) of what each line of code is doing in its execution. For example, we have a code snippet of the bubble sort algorithm.
Given the array called vector that contains the elements [6, 5, 3, 1, 8, 7, 2, 4], the bubble algorithm code is presented to order the elements of the array from smallest to largest. With an explanation like a desktop test how the variables take the values in the first iteration of the cycle (see Fig. 2).

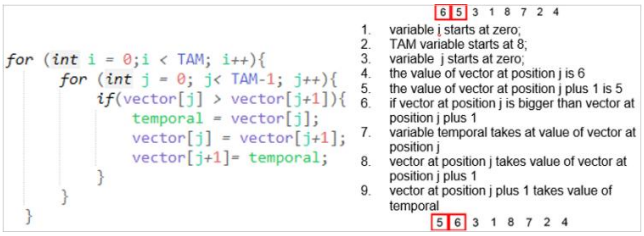

Fig. 2. Desktop test example: The Bubble Algorithm Code.

2. Present text and visualizations separately:
   a. Integrate text and visualizations. An example is given in Table 2.

Table 2. An example of integrate theoretical and graphic information.

Theoretical explanation	Code	First iteration: Explanation and Graphic visualization

Bubble sort algorithm: Each iteration com-parisons are made of one element(element ”j”) with the previous element(element ”j-1”) , and if element ”j” is less than element ”j-1” they are swapped. In this way, the largest element of the array is raised to the end of it.	for(int i = 0; i < TAM; i++){
	for(int j = 0; j < TAM-1; j++){
	if(vector[j] > vector[j]+1){
	temporal = vector[j];
	vector[j] = vector[j]+1;
	vector[j]+1 = temporal;
	}
	}
	}
		6 5 3 1 8 7 2 4
		1. Variable i start at zero.
		2. TAM variable starts at 8.
		3. Variable j starts at zero.
		4. Value of vector at position j is 6.
		5. Value of vector at position j + 1 is 5.
		6. If vector at position j is bigger than vector at position j + 1
		7. Variable temporal takes at value of vector at position j
		8. vector at position j + 1 takes value of temporal
		5 6 3 1 8 7 2 4

3. Shows too much transient information: Transient information refers to information that is only available for a few seconds to the student. Some tips for studying this type of material are below:
   a. Control the pace of visualizations, that is, when watching videos, presentations, or animations, pause and repeat until you understand the concepts.
   b. Preferably, transfer this important transitory information to non-transitory information, that is, take notes, applying the advice from the two previous points.

Summary of the self-management guide:

- Use integrated code comments rather than separate explanations.
- Eliminate redundant information.
- Present concepts progressively without repetition.
- Use visual elements like diagrams to show memory allocation.
- Provide concise, focused explanations directly adjacent to relevant code.
- Use meaningful variable names and examples that don't require extensive explanation.
- Persistent visual references.
- Learner-controlled pacing.
- Cumulative code building.
- Static summaries alongside dynamic content.

To implement the study instructional materials on C++ data structures were developed with intentional redundancy and split-attention issues. For the transient information effect, students were directed to online video tutorials. Some examples of redundancy and divided attention issues applied to the material provided to students are:

Redundancy Issues:

- The introduction repeats the same information multiple times.
- Code explanations are overly verbose, explaining obvious elements like line numbers and basic syntax.

- The summary restates information already covered multiple times.
- Multiple sections say the same thing about basic concepts and definitions.

Split-attention Issues:

- Code and explanations are separated, forcing learners to mentally connect distant text.
- Line-by-line explanations require constant switching between code and text.
- The step-by-step breakdown forces readers to jump back and forth between numbered steps and the actual code.
- Visual attention is scattered across multiple sections rather than integrated.

Then the learning treatment was applied through the self-management guide and the collective working memory effect explained in step 2. While cognitive load was measured using the CS Cognitive Load Component Survey [35] because, methods for measuring cognitive load are divided into objective and subjective. Objective measures include physiological data such as eye tracking, heart rate, and neural activity. Subjective measures ask students to report their perceived mental effort. Computational education research has explored various objective methods, but these methods are intrusive and difficult to differentiate between types of cognitive load. Therefore, subjective methods have been more widely used, one of the most cited being Morrison et al. specifically for programming education. Factor analysis confirmed internal reliability and discriminant validity, establishing it as a standard tool for assessing cognitive load in programming learning contexts [36].

To evaluate learning outcomes, a recall test for declarative knowledge was developed in conjunction with the instructor. For procedural knowledge a comprehension test with real-world problem-solving was developed. Self-management skills were evaluated using a scale that measured students' adherence to the guide's principles and compared their created materials with instructor-generated content.

3.2. Calculating the sample size and conducting a power analysis

To ascertain if the sample sizes were sufficient for identifying significant effects, a power analysis was carried out. We expected a medium impact size (Cohen's d = 0.5) for the comprehension test, which is the main outcome of interest in knowledge transfer, based on other studies on CLT interventions in programming education [4, 8, 17, 18, 20, 22, 24, 25, 26]. Using G*Power 3.1.9.7, with the following parameters:

$$effect\_size(d) = 0.5(medium\_effect) \quad (1)$$

$$\alpha = 0.05(two - tailed) \quad (2)$$

$$Power(1 - \beta) = 0.80 \quad (3)$$

The necessary sample size was determined to be n = 52 per group for non-parametric tests (Wilcoxon) and n = 64 per group for independent samples t-tests. The current investigation is limited by the statistical power of roughly 0.65 for identifying medium effects, which is provided by our achieved sample sizes (Experimental: n = 36, Control: n = 41).

Since the study reflects a naturalistic classroom context where sample sizes are limited by course registration, we went forward with it despite the lower than ideal power. Despite the sample size limits, post-hoc power analysis verified that the comprehension test's substantial difference ($p < .001$) was robust.

3.3. Step 2: Experimental design

A quasi-experimental study was conducted with two groups of second-semester Computer Systems Engineering students at the Autonomous University of Aguascalientes: an experimental group (Exp, n=36) and a control group (Cont, n=41), with the same instructor. The groups were assigned based on the instructor's prior knowledge of the groups, as both groups had previously taken an introductory programming course. The same instructor continued with the next introductory programming course. At the beginning of the experiment, both groups had very similar grades (Exp 6.6 and Cont 6.3) according to the results of standardized and calibrated exams administered by the Programming Department of the Electrical Systems Department of the Autonomous University of Aguascalientes. Both groups had a homogeneous prior knowledge base (novices).

The experimental treatment combined two CLT effects: Self-Management and Collective Working Memory effects. Students in the experimental group were guided through a Self-Management effect to develop their skills in preparing study materials and applying the Collective Working Memory effect the treatment was implemented in teams. The detailed phases of the quasi-experimental design are presented in Table 3.

As shown in Table 3, a 5-phase quasi-experimental design was implemented for the application of the Self-Management and Collective Working Memory, the phases of the quasi-experiment are detailed below:

Table 3. Quasi-experimental design phases: Application of the Self-Management and Collective Working Memory effects in programming.

Phase	Experimental group (Exp)	Control group (Cont)
1: Learning treatment	Application of Self-Management and Collective Working Memory effects	Traditional classes by the instructor
2: Review/Study	Study your own material individually at home	Study your notes, examples, exercises in a traditional way
3: Recall test	Apply a questionnaire of the basic and important concepts (Individual)	Apply a questionnaire of the basic and important concepts (Individual)
4: Comprehension test	Problems to solve on the topic of structures (Teams)	They will solve the problems individually during class (Individual)
5: Cognitive load	Apply CS Cognitive Load Component Survey	Apply CS Cognitive Load Component Survey

1. Phase 1: Learning treatment
   Provide support material and create study material (experimental group):
   - The self-management guide is provided to students. Following this, the instructor explains that this guide was developed to support them in improving the design of their own study materials, based on materials with poor instructional design. This guide encourages the development of self-management skills.
   - Instructional material with poor instructional design based on C++ data structures developed with intentional redundancy and split-attention issues was delivered to students for the purpose of creating their own study material, eliminating design issues by following the instructions in the provided self-management guide. The instructor always accompanied the process, providing feedback as required by the students.
   - To apply the collective working memory effect, groups of three members were formed, distributing the students heterogeneously based on their previous grades on official exams, to balance the teams and comply with the guidelines of the effect.
   - The working groups developed their own material based on the self-management guide provided. The process was developed in two classes, which were supervised and always supported by the instructor.
   - Traditional classes by the instructor (control group): The teacher presents and explains the concepts to be learned, using traditional strategies such as presentations, sample exercises, etc. Students take notes on the topics.

2. Phase 2: Review/Study
   - The experimental group uses their created material to study the topics prior to the recall and comprehension tests. The study or review activity was carried out individually.
   - The control group studies your notes, examples, exercises taken in classes in a traditional way.

3. Phase 3: Recall test
   - To assess students' acquisition of basic concepts, an individual online test ('Recall Test') was administered using Microsoft Forms for both groups (Exp and Cont). The test was developed in conjunction with the instructor. This assessment focused on fundamental concepts related to data structures. Sample questions from the test are presented below:
     What are data structures?
     What is the syntax for defining a structure?
     How to initialize a structure. Please explain in detail and provide an example.
     What is a nested structure? Provide an example of nested structure definition, variable declaration, and data access in nested structures.

4. Phase 4: Comprehension test
   - A 'Comprehension Test' consisting of practical problem-solving tasks was administered to assess knowledge transfer.
   - For Exp group the assessment was conducted in three-member teams applying Collective Working Memory effect, because the description of this effect mentions that it helps develop the knowledge transfer skill.
   - The application format was through a game-timed competitive format, where teams competed to solve problems accurately and efficiently.
   - The first five teams submitting correct solutions received bonus points.
   - This competitive structure was designed to strengthen individual knowledge transfer through collaborative problem-solving skills. A representative problem from the test is presented below:
     In high school, it is required to issue some reports about the N students in a group of students. Among the most important data are id, name, semester, group, subject. Where the subject is composed of the

following fields: subject key, name, and partial grades (3 in total). The following reports are requested to be generated:

- *Show a list with the id and name of the students with an average greater than 8.5.*
- *Show a list of the students who did not pass the subject with the ABC key, considering 7 as a sufficient grade to pass.*
- *Show a list of the students who have the highest grade (between 9 and 10).*
- *Show the individual average of each student in the Programming subject.*
- *It is important to create a function that allows the filling of the array of size N indicated by the user.*
- *Points to be evaluated in this exercise: nested structures, structure type arrays, parameter passing, functions, enumerations.*

- The control group solved the problems individually.

5. Phase 5: Cognitive load measurement
   - Cognitive load was assessed using the CS Cognitive Load Component Survey [35].
   - The survey consists of ten items rated on an 11-point scale ranging from 0 (not at all the case) to 10 (completely the case).
   - Data collection was conducted through an online form developed using Microsoft Forms.

Both the experimental and control groups were taught by the same instructor, who was familiar with them from an earlier introductory course. As a result, his role in creating the self-management guide and materials that addressed issues of redundancy, divided attention, and transient information was vital. In a similar vein, the instructor offered direction and feedback on any queries they had regarding the quasi-experimental procedure. Participation by students in the quasi-experiment also depended on this.

4. Data Analysis and Results

For data analysis, the dependent variables were scores on the recall test (declarative knowledge), the comprehension test (procedural knowledge/knowledge transfer) and self-management skills. The independent variables were group, intrinsic cognitive load (IL), extrinsic cognitive load (EL) and relevant load (GL). And finally, potential control/confounding variables were prior knowledge performance (Exp = 6.6, Cont = 6.3) and individual versus team assessments (Recall individual, Comprehension grupal vs individual).

4.1. Confounding variable and statistical control analysis

Possible confounding factors that might affect the association between the Cognitive Load Theory (CLT) interventions and the acquired learning outcomes were uncovered and assessed before the primary analysis was developed. The participants' prior academic performance (Prior_P) was one of the confounding variables that was found; the experimental group's average score was 6.6, while the control group was 6.3. This could skew the results in favor of the group that performed better at the beginning. Similarly, differences were found in the assessment format and instructional material (Treatment) used, Self-Management effect and the individual versus group modality, which may alter the measurement settings and impact the results' comparability. Lastly, differences in intrinsic load (IL), extrinsic load (EL), and germane load (GL) levels between the groups were found to indicate the presence of differential cognitive load. This could jeopardize the study's internal validity by introducing uncontrollable factors that affect the participants' cognitive processing.

4.2. Statistical analysis

The analysis compared learning outcomes of recall and comprehension test scores between the experimental and control groups for C++ data structures topics. Table 4 presents descriptive statistics, which show that the experimental group achieved mean scores of 6.12 and 8.60 on the recall and comprehension tests, respectively, while the control group obtained mean scores of 6.28 (recall) and 6.59 (comprehension).

Table 4. Descriptive statistics: Recall and Comprehension tests of groups participating in the study.

Group	Test	N	Minimum	Maximum	Mean
Exp	Recall	36	2.1	8.8	6.12
	Comprehension	36	3.5	10.0	8.60
Cont	Recall	41	2.5	9.2	6.28
	Comprehension	41	0.0	10.0	6.59

4.2.1. Multiple regression analysis

For both dependent variables (recall test and comprehension test), multiple regression models were conducted to simultaneously control several confounding variables. Tables 5 and 6 show the results of the regression model for the

dependent variable Recall_test. Where R^2 is observed with a value of 0.169 (16.9% of the variance explained), adjusted R^2 of 0.107 and a standard error of 1.361. With one significant coefficient, Prior_P, with a p-value of .011. The IL, EL and GL variables were not significant. Tables 7 and 8 show the results of the model for the dependent variable Comprehension_test. The results show R^2 0.165 (16.5% of the variance explained), adjusted R^2 de 0.102 and a standard error of 2.878. The significant coefficient is the variable Treatment with a p-value of .002.

Table 5. Model summary for recall test.

Model	R	R Square	Adjusted R Square	Std. Error Estimate
1	.411[a]	.169	.107	1.361

a. Predictors: (Constant), Treatment, EL, Prior_P, IL, GL.

Table 6. Coefficients[a]: recall test.

Model	Unstandardized Coefficients		Standardized Coefficients	t	Sig.
	B	Std. Error	Beta		
1 (Constant)	5.978	1.188		5.033	.000
IL	-.025	.086	-.037	-.297	.767
EL	-.135	.087	-.245	-1.551	.126
GL	-.045	.118	-.060	-.377	.707
Prior_P	.171	.065	.323	2.609	.011
Treatment	-.292	.343	-.101	-.851	.398

The SPSS tool was used to apply the regression model and generate the results. Recall test model and Comprehension test model are shown below:

$$Recall\_Test = \beta\_0 + \beta\_1\,(Treatment) + \beta\_2\,(Prior\ P) + \beta\_3\,(IL) + \beta\_4\,(EL) + \beta\_5\,(GL) + \varepsilon \quad (4)$$

$$Comprehension\_Test = \beta\_0 + \beta\_1\,(Treatment) + \beta\_2\,(Prior\ P) + \beta\_3\,(IL) + \beta\_4\,(EL) + \beta\_5\,(GL) + \varepsilon \quad (5)$$

Table 7. Model summary for comprehension test.

Model	R	R Square	Adjusted R Square	Std. Error Estimate
1	.406[a]	.165	.102	2.878

a. Predictors: (Constant), Treatment, EL, Prior_P, IL, GL.

Table 8. Coefficients[a]: comprehension test.

Model	Unstandardized Coefficients		Standardized Coefficients	t	Sig.
	B	Std. Error	Beta		
1 (Constant)	3.422	2.511		1.363	.178
IL	-.036	.181	-.025	-.201	.841
EL	.167	.184	.144	.908	.367
GL	.336	.250	.216	1.344	.183
Prior_P	.053	.138	.047	.382	.704
Treatment	2.312	.725	.380	3.187	.002

4.2.2. Simple mediation analysis

A simple mediation analysis was performed to assess whether any of the cognitive loadings (IL, EL and GL) mediate the relationship between treatment and comprehension test performance.

$$Treatment \rightarrow Cogni\ \ Load\ Component \rightarrow Comprehension\ test \quad (6)$$

We assessed whether intrinsic load (IL) mediates the relationship between group assignment and performance on the comprehension test. Python was used for the analysis, as shown below:

Model predicting mediator (M) from independent variable (X):

$$model\_m = smf.ols(f'M \sim X', data = df).fit() \quad (7)$$

Model predicting dependent variable (Y) from independent variable (X) and mediator (M):

$$model\_y = smf.ols(f'Y\ X + M', data = df).fit() \quad (8)$$

Total effect (direct effect + indirect effect) is the coefficient of X in a model predicting Y from X:

$$model\_total = smf.ols(f\text{^}'Y\ X, data = df).fit()total\_effect = model\_total.params[X] \quad (9)$$

Direct effect is the coefficient of X in the model predicting Y from X and M:

$$direct\_effect = model\_y.params[X] \quad (10)$$

Indirect effect is the product of the coefficient of X in model_m and the coefficient of M in model_y:

$$indirect\_effect = model\_m.params[X] * model\_y.params[M] \quad (11)$$

Mediation Analysis Summary:

- Total Effect: 2.01
- Direct Effect: 2.26
- Indirect Effect: -0.05
- P-value for Treatment - IL path: 0.146
- P-value for IL - Comprehension_test path: 0.649
- P-value for Direct effect (Treatment - Comprehension_test): 0.002

The results show the total effect of "Treatment" on "Comprehension_test" was calculated as 2.01. The direct effect of "Treatment" on "Comprehension_test", controlling for "IL", was 2.27. The indirect effect of "Treatment" on "Comprehension_test" through "IL" was -0.05. The direct effect of "Treatment" on "Comprehension_test" was found to be statistically significant (p = 0.002). Based on the significance of the individual paths, there is no strong evidence for a statistically significant indirect effect (mediation) of "IL" in the relationship between "Treatment" and "Comprehension_test".

We assessed whether intrinsic load (EL) mediates the relationship between group assignment and performance on the comprehension test. EL is not a significant mediator in the relationship between 'Treatment' and 'Comprehension_test'. Results mediation analysis summary below:

- Total Effect: 2.21
- Direct Effect: 2.22
- Indirect Effect: -0.01
- P-value for Treatment - IL path: 0.0016
- P-value for IL - Comprehension_test path: 0.960
- P-value for Direct effect (Treatment - Comprehension_test): 0.0017

Similarly, mediation was applied with the GL variable. Based on the results of this regression analysis, there is no evidence that the variable GL mediates the relationship between Treatment and Comprehension_test. For mediation to occur, one would generally expect a significant effect of the predictor (Treatment) on the mediator (GL) and a significant effect of the mediator (GL) on the dependent variable (Comprehension_test) when controlling for the predictor. In this case, the effect of Treatment on GL and the effect of GL on Comprehension_test are not statistically significant.

4.2.3. Cognitive load analysis and relationships

Important trends in the interactions and effects of various load types on learning outcomes were found by the cognitive load assessments. The cognitive load assessments for both groups are displayed in Tables 9 and 10, but a more thorough examination reveals important connections between the different types of burden and how they affect learning performance. The questionnaire assessed three types of cognitive load: Intrinsic Load (IL), Extraneous Load (EL), and Germane Load (GL).

For the control group, Table 10 presents the cognitive load measurements after completing the quasi-experiment. It shows a medium-high level of intrinsic load and a low level of extraneous load, and a high germane cognitive load. A Pearson correlation test was performed for the experimental group with the cognitive load measurement and the grades of recall and comprehension tests. Table 11 shows the results, which do not show a significant correlation between the grades and the cognitive load, however, it shows a significant negative correlation between the germane load and extraneous load.

Table 9. Results of the measurement of cognitive load: experimental group.

Question	Avg	σ	Factor Avg
1	6.56	1.76	IL = 6.16
2	6.44	2.00	
3	5.47	2.51	
4	3.43	2.67	EL = 3.27
5	3.53	2.82	
6	3.43	2.51	
7	7.00	2.12	GL = 6.91
8	6.80	2.20	
9	6.87	1.69	
10	6.87	2.16	

Table 10. Results of the Measurement of Cognitive Load: Control Group

Question	Avg	σ	Factor Avg
1	5.83	2.41	IL = 5.43
2	5.29	2.59	
3	5.17	2.41	
4	3.10	3.28	EL = 2.93
5	2.76	2.70	
6	2.93	2.99	
7	7.56	2.07	GL = 7.46
8	7.59	1.86	
9	7.34	2.46	
10	7.37	2.47	

Table 11. Correlations Experimental Group

		1	2	3	4	5
1. Comprehension test	Pearson Correlation Sig. (2-tailed)	1				
2. Recall test	Pearson Correlation Sig. (2-tailed)	.074 .689	1			
3. Intrinsic Load	Pearson Correlation Sig. (2-tailed)	-.246 .174	-.146 .426	1		
4. Extraneous Load	Pearson Correlation Sig. (2-tailed)	.094 .610	-.185 .312	.062 .735	1	
5. Germane Load	Pearson Correlation Sig. (2-tailed)	.003 .985	.286 .112	-.166 .364	-.733** .000	1

**. Correlation is significant at the 0.01 level (2-tailed).

Table 12 shows the results of the Pearson correlation test for the control group. There is no significant correlation between grades and cognitive load; however, there is a significant negative correlation between germane load, extraneous load, and intrinsic load. An independent samples t-test was performed on the cognitive load variables of both groups. Table 13 shows the results, which indicate that there is no significant statistical difference between the cognitive load values of the experimental and control groups.

Table 12. Correlations Control Group.

		1	2	3	4	5
1. Comprehension test	Pearson Correlation Sig. (2-tailed)	1				
2. Recall test	Pearson Correlation Sig. (2-tailed)	-.183 .252	1			
3. Intrinsic Load	Pearson Correlation Sig. (2-tailed)	.009 .953	-.228 .151	1		
4. Extraneous Load	Pearson Correlation Sig. (2-tailed)	-.061 .705	-.303 .054	.477** .002	1	
5. Germane Load	Pearson Correlation Sig. (2-tailed)	.212 .183	.158 .325	-.352* .024	-.651** .000	1

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

Experimental group load relationships:

- Intrinsic Load (IL = 6.16): Medium-high level, reflecting the inherent complexity of C++ data structures.
- Extraneous Load (EL = 3.27): Low level, indicating successful reduction of unnecessary cognitive burden.
- Germane Load (GL = 6.91): Medium-high level, suggesting active schema construction.

Table 13. Independent Samples Test of cognitive load variables for experimental and control groups

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Diff.	Std. Error Diff.	95% Confidence Interval	
									Lower	Upper
Intrinsic Load	Equal variances assumed	2.35	.130	1.47	71	.15	.73	.49	-.26	1.71
	Equal variances not assumed			1.52	70.8	.13	.73	.46	-.22	1.68
Extraneous Load	Equal variances assumed	1.91	.171	.55	71	.58	.34	.62	-.89	1.58
	Equal variances not assumed			.57	70.5	.57	.34	.60	-.87	1.56
Germane Load	Equal variances assumed	.04	.840	-1.21	71	.23	-.56	.46	-1.47	.36
	Equal variances not assumed			-1.23	68.5	.22	-.56	.46	-1.46	.35

Control group load relationships:

- Intrinsic Load (IL = 5.43): Medium level, slightly lower than experimental group.
- Extraneous Load (EL = 2.93): Low level, like experimental group.
- Germane Load (GL = 7.46): High level, higher than experimental group.

Germane Load and Extraneous Load were significantly correlated negatively in both groups (Exp: r = -.733, p<.001; Cont: r = -.651, p <.001). This result lends credence to the "new CLT" framework, which defines relevant load as the effective utilization of working memory resources after taking into consideration both intrinsic and extrinsic demands. Implications for learning outcomes below:

- Recall test: The comparable group performance (Exp: 6.13, Cont: 6.28, p =.841) indicates that load type differences within the reported ranges do not significantly affect the acquisition of basic knowledge.
- Comprehension test: Despite having a greater intrinsic load, the experimental group performed better (Exp: 8.80, Cont: 6.59, p < .001), which implies that the Self-Management and Collective Working Memory processes improved knowledge transfer capacities.

4.2.4. Self-Management analysis

The rubric evaluated students' self-management ability through their created study materials, based on the self-management guide. Scores ranged from 0 to 10 (Table 14). Table 15 presents the descriptive statistics of student grades, while Fig. 3 displays the score distribution, showing most scores above 7.

Table 14. Rubric to Rate the Study Material Based on the Self-Management Effect

Category	Aspects to evaluate		
Redundancy effect	Complement with visualizations such as drawings or concept maps, alluding to the cited text.	Eliminate nonessential or redundant information, generate textual or visual summaries, and indicate essential information by highlighting texts or images.	Accompany the text of the code with textual or graphic explanations (desktop test) of what each line if code is doing in its execution.
Split attention effect	Integrate text and visualizations in the same visual area		
Content	Complete	Organized	Understandable

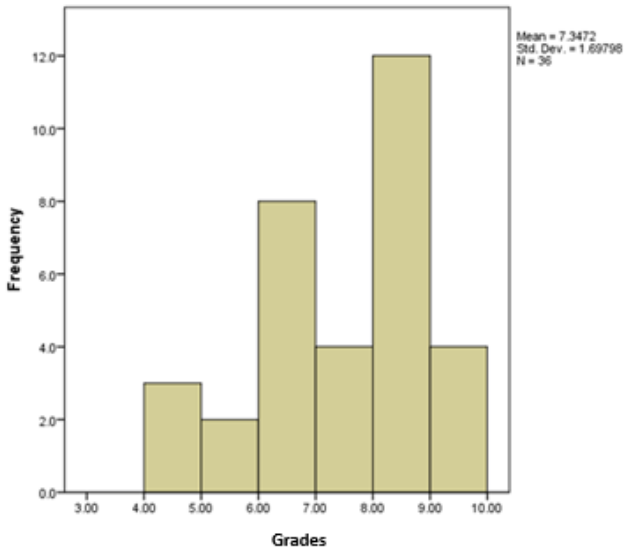

Fig. 3. Histogram of rubric Self-Management material observations of experimental group.

Table 15. Descriptive Statistics of Rubric Self-Management Material for Experimental Group

	N	Minimum	Maximum	Mean	Std. Deviation
Grade Self-Management skills	36	4.0	10.0	7.34	1.70
Valid N (listwise)	36				

5. Discussion and Conclusion

This study examined how cognitive load theory's Self-Management effect impacts learning data structures in programming. It investigates the development of self-management skills, incorporating the Collective Working Memory effect. Learning effectiveness was assessed through recall of theoretical concepts, problem-solving transfer exams, and self-management ability through a rubric based on guidelines from the self-management guide, where the scores ranged from 0 to 10.

Results showed comparable performance between experimental and control groups on the basic knowledge (Recall) test. However, the experimental group achieved significantly higher scores on the knowledge transfer (Comprehension) test, confirmed by t-test analysis. These findings align with previous research on the Self-Management effect [18, 19, 20, 21, 22, 37], suggesting that improved transfer test performance stems from enhanced self-management developed through CLT's Self-Management and Collective Working Memory effects; however, this was not the case in the understanding of theoretical core concepts.

Multiple regression analysis confirms that the treatment effect on the comprehension test is fair, even after controlling some confounding variables. Prior performance acts as a significant confounder in both models, justifying its inclusion in the analyses. The level of intrinsic load moderates the treatment effect, suggesting that the benefits are greater for students facing greater cognitive complexity. No evidence of mediation through extrinsic load was found, suggesting that other mechanisms explain the treatment effect.

Post-experiment cognitive load measurements showed higher intrinsic and extraneous loads in the experimental group compared to the control group. This increased cognitive load likely resulted from students' additional mental effort in creating study materials that avoided redundancy, split-attention, and transient information problems, using the provided guidelines. Although the experimental group had higher intrinsic and extraneous cognitive loads, there were no statistically significant differences in the cognitive loads between the experimental and control groups. Nevertheless, we hypothesize that the additional effort of the experimental group in developing self-management skills ultimately produced higher-quality study materials and better learning outcomes.

The study's findings show clear trends in how different forms of learning are affected by cognitive load. Recall tests measuring declarative knowledge revealed no significant relationships between any kind of cognitive load and performance, indicating that basic concept memory is largely unaffected by changes in cognitive load within the ranges that were tested. The identical intrinsic loads (6.16 vs. 5.43) across the two groups, which lead to equivalent recall ability, could account for this lack of variances.

On the other hand, a seemingly contradictory pattern emerged in the assessment of procedural knowledge through comprehension tests: despite a higher intrinsic load, the experimental group’s comprehension performance was noticeably better. The self-management process increased working memory efficiency and enabled the experimental group to produce study materials that were designed to improve their application of procedural knowledge explains this seeming paradox.

According to new CLT [5,6,18], this counterintuitive effect allowed the experimental group to convert a higher intrinsic load into better performance through cognitive load optimization by increasing students understanding of the relationships between highly related programming concepts. Self-Management enhanced intrinsic load (6.16 vs. 5.43) by increasing their awareness of item interactivity. But at the same time, this procedure increased working memory efficiency, enabling students to cultivate better cognitive resource management techniques and promoting better schema creation through better knowledge structure organization.

The collaborative approach of the experimental group amplified these benefits through the collective working memory effect, where teams distributed cognitive processing among their members, integrated complementary knowledge by leveraging different levels of expertise, and reduced individual extrinsic load through coordination that helped identify and eliminate redundant information processing, thus creating a more efficient and effective learning environment.

The rubric evaluation of student-created learning materials showed positive results, aligning with students' performance on both Recall and Comprehension tests.

Overall, these findings support the Self-Management effect claims and CLT-based knowledge management, particularly with guided feedback. Students who self-managed their instructional materials demonstrated improved knowledge transfer to real-world problems, even if this initially involves a greater cognitive load, but it does not necessarily affect student performance. Future research should explore CLT-compatible effects combination and refine the guided self-management process. While our study contributes to understanding the Self-Management effect, additional research is needed to optimize its application and outcomes.

6. Limitations and Future Work

Future research should explore CLT-compatible effects combinations and refine the guided self-management process. While our study contributes to understanding the Self-Management effect, additional research is needed to optimize its application and outcomes to other areas of computer science learning.

The positive results remain valid despite the lack of prior knowledge assessments. The naturalistic approach reflects real-world conditions, where such data is often unavailable. The intervention's success across diverse participants indicates its effectiveness, making the findings robust and applicable. Therefore, the absence of baseline assessments does not diminish the outcomes.

The study did not control the potential use of undisclosed study materials beyond the self-generated content, which could have influenced the validity of the findings. Additionally, despite the instructor's supervision, there could have been instances of response copying in the tests that required the elimination of certain samples despite the implementation of control measures. And finally, the quasi-experimental design using pre-established cohorts limits the generalizability of the results. Further replication could enhance generalizability.

The sample sizes in this study (Control: n = 41, Experimental: n = 36) are below the ideal cutoff point for identifying medium effect sizes with sufficient statistical power (0.80). Our study's power to detect medium effects was only 65%, according to power analysis. This restricts the generalizability of null findings, especially for the recollection test, where no significant differences were found. Notwithstanding the limitations of sample size, the comprehension test revealed substantial effects ($p < .001$) that were adequately robust. Additionally, the small sample size limits the capacity to do more complex analyses, including:

- Subgroup analyses based on prior programming experience.
- Multilevel modeling to account for classroom-level effects.
- Interaction effects between different student characteristics and the intervention.

Another limitation is unobserved variables. Possible unmeasured confounding variables (motivation, specific prior experience, time, cognitive and metacognitive strategies) could influence the results. Nested structure: the data have nested structure (students in teams) that was not fully modeled due to sample size limitations. Although the analyses are fairer, causal inference remains limited by the quasi-experimental design.

To attain sufficient statistical power and facilitate more thorough analysis, future studies should strive for higher sample numbers (minimum n = 64 per group). As well as identifying all possible confounding variables, expanding the study to more than one institution, greater control over the experiment.

References

[1] E. C. K. Cheng, "Knowledge transfer strategies and practices for higher education institutions," VINE Journal of Information and Knowledge Management Systems, vol. 51, no. 2, pp. 288–301, 2020, doi: 10.1108/VJIKMS-11-2019-0184.

[2] H. W. Fennell, J. A. Lyon, A. Madamanchi, and A. J. Magana, "Toward computational apprenticeship: Bringing a constructivist agenda to computational pedagogy," Apr. 01, 2020, John Wiley and Sons Inc. doi: 10.1002/jee.20316.

[3] C. Izu and C. Mirolo, "Learning Transfer in Novice Programmers: A Preliminary Study," in Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE, Association for Computing Machinery, Jun. 2021, pp. 178–184. doi: 10.1145/3430665.3456336.

[4] Y. T. Lin, M. K. C. Yeh, and S. R. Tan, "Teaching Programming by Revealing Thinking Process: Watching Experts' Live Coding Videos With Reflection Annotations," IEEE Transactions on Education, vol. 65, no. 4, pp. 617–627, Nov. 2022, doi: 10.1109/TE.2022.3155884.

[5] J. Sweller, J. J. G. van Merri ënboer, and F. Paas, "Cognitive Architecture and Instructional Design: 20 Years Later," Educ Psychol Rev, vol. 31, no. 2, pp. 261–292, 2019, doi: 10.1007/s10648-019-09465-5.

[6] A. B. H. de Bruin et al., "Synthesizing Cognitive Load and Self-regulation Theory: a Theoretical Framework and Research Agenda," Educ Psychol Rev, vol. 32, no. 4, pp. 903–915, 2020, doi: 10.1007/s10648-020-09576-4.

[7] A. Eitel, T. Endres, and A. Renkl, "Self-management as a Bridge Between Cognitive Load and Self-regulated Learning: the Illustrative Case of Seductive Details," Educ Psychol Rev, vol. 32, no. 4, pp. 1073–1087, 2020, doi: 10.1007/s10648-020-09559-5.

[8] S. Zhang, B. B. de Koning, and F. Paas, "Finger pointing to self-manage cognitive load in learning from split-attention examples," Appl Cogn Psychol, vol. 36, no. 4, pp. 767–779, 2022, doi: 10.1002/acp.3961

[9] J. Janssen and P. A. Kirschner, "Applying collaborative cognitive load theory to computer-supported collaborative learning: towards a research agenda," Educational Technology Research and Development, vol. 68, no. 2, pp. 783–805, 2020, doi: 10.1007/s11423-019-09729-5.

[10] J. Sweller, J. J. G. van Merriënboer, and F. Paas, "Cognitive Architecture and Instructional Design," Educ Psychol Rev, vol. 10, no. 3, pp. 251–295, 1998.

[11] J. Sweller, P. Ayres, and K. S., Cognitive Load Theory. 2011. [Online]. Available: http://www.springer.com/series/8640.

[12] D. C. Geary, "Evolutionary educational psychology.," in APA educational psychology handbook, Vol 1: Theories, constructs, and critical issues., in APA handbooks in psychology®. , Washington, DC, US: American Psychological Association, 2012, pp. 597–621. doi: 10.1037/13273-020.

[13] R. C. Atkinson and R. M. Shiffrin, "Human Memory: A Proposed System and its Control Processes11This research was supported by the National Aeronautics and Space Administration, Grant No. NGR-05-020-036. The authors are indebted to W. K. Estes and G. H. Bower who provided many valuable sugge," vol. 2, K. W. Spence and J. T. B. T.-P. of L. and M. Spence, Eds., Academic Press, 1968, pp. 89–195. doi: https://doi.org/10.1016/S0079-7421(08)60422-3.
[14] C.-C. Carbon and S. Albrecht, "Bartlett's schema theory: The unreplicated 'portrait d'homme' series from 1932," Quarterly Journal of Experimental Psychology, vol. 65, no. 11, pp. 2258–2270, Nov. 2012, doi: 10.1080/17470218.2012.696121.
[15] R. Duran, A. Zavgorodniaia, and J. Sorva, "Cognitive Load Theory in Computing Education Research: A Review," ACM Transactions on Computing Education, vol. 22, no. 4, 2022, doi: 10.1145/3483843.
[16] J. H. Berssanette and A. C. de Francisco, "Cognitive Load Theory in the Context of Teaching and Learning Computer Programming: A Systematic Literature Review," IEEE Transactions on Education, vol. 65, no. 3, pp. 440–449, 2022, doi: 10.1109/TE.2021.3127215.
[17] F. Paas and J. J. G. van Merriënboer, "Cognitive-Load Theory: Methods to Manage Working Memory Load in the Learning of Complex Tasks," Curr Dir Psychol Sci, vol. 29, no. 4, pp. 394–398, 2020, doi: 10.1177/0963721420922183.
[18] S. T. M. Sithole, P. Chandler, I. Abeysekera, and F. Paas, "Benefits of guided self-management of attention on learning accounting," J Educ Psychol, vol. 109, no. 2, pp. 220–232, 2017, doi: 10.1037/edu0000127.
[19] J. C. Castro-Alonso, B. B. de Koning, L. Fiorella, and F. Paas, "Five Strategies for Optimizing Instructional Materials: Instructor- and Learner-Managed Cognitive Load," Educ Psychol Rev, vol. 33, no. 4, pp. 1379–1407, 2021, doi: 10.1007/s10648-021-09606-9.
[20] B. B. de Koning, G. Rop, and F. Paas, "Learning from split-attention materials: Effects of teaching physical and mental learning strategies," Contemp Educ Psychol, vol. 61, no. April, p. 101873, 2020, doi: 10.1016/j.cedpsych.2020.101873.
[21] B. B. de Koning, G. Rop, and F. Paas, "Effects of spatial distance on the effectiveness of mental and physical integration strategies in learning from split-attention examples," Comput Human Behav, vol. 110, no. April, 2020, doi: 10.1016/j.chb.2020.106379.
[22] K. Roodenrys, S. Agostinho, S. Roodenrys, and P. Chandler, "Managing one's own cognitive load when evidence of split attention is present," Appl Cogn Psychol, vol. 26, no. 6, pp. 878–886, 2012, doi: 10.1002/acp.2889.
[23] M. Beege, S. Schneider, S. Nebel, J. Zimm, S. Windisch, and G. D. Rey, "Learning programming from erroneous worked-examples. Which type of error is beneficial for learning?," Learn Instr, vol. 75, p. 101497, 2021, doi: 10.1016/j.learninstruc.2021.101497.
[24] P. Sands, "Addressing cognitive load in the computer science classroom," ACM Inroads, vol. 10, no. 1, pp. 44–51, 2019, doi: 10.1145/3210577.
[25] R. Zhi, M. Chi, T. Barnes, and T. W. Price, "Evaluating the effectiveness of parsons problems for Block-based programming," ICER 2019 - Proceedings of the 2019 ACM Conference on International Computing Education Research, pp. 51–59, 2019, doi: 10.1145/3291279.3339419.
[26] C. Arevalo-Mercado, E. Muñoz-Andrade, H. Cardona Reyes, and M. Romero-Juárez, "Applying Cognitive Load Theory and The Split Attention Effect to Learning Data Structures," IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 18(1), pp. 107–113, 2023, doi: 10.1109/RITA.2023.3250580.
[27] J. Sweller, "Cognitive load theory and educational technology," Educational Technology Research and Development, vol. 68, no. 1, pp. 1–16, 2020, doi: 10.1007/s11423-019-09701-3.
[28] R. E. Vieira, C., Magana, A. J., Falk, M. L., and Garcia, "Writing in-code comments to self-explain in computational science and engineering education.," ACM Transactions on Computing Education (TOCE), vol. 17, no. 4, pp. 1–21, 2017, doi: 10.1145/3058751.
[29] Geela Venise Firmalo Fabic, Antonija Mitrovic and Kourosh Neshatian "Investigating the Effectiveness of Menu-Based Self-explanation Prompts in a Mobile Python Tutor," Aied 2017, vol. 1, no. August 2018, pp. 498–501, 2017, doi: 10.1007/978-3-319-61425-0_49.
[30] C. W. Yen and T. I. Wang, "Using Self-Explanation and Ontology for Providing Proper Feedbacks in a Programming Environment," In 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), pp. 585–590, 2017, doi: 10.1109/IIAI-AAI.2017.136.
[31] T. W. Price, J. J. Williams, J. Solyst, and S. Marwan, "Engaging Students with Instructor Solutions in Online Programming Homework," In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, pp. 1–7, 2020, doi: 10.1145/3313831.3376857.
[32] V. Winter, M. Friend, M. Matthews, B. Love, and S. Vasireddy, "Using Visualization to Reduce the Cognitive Load of Threshold Concepts in Computer Programming," Proceedings - Frontiers in Education Conference, FIE, vol. 2019-Octob, pp. 1–9, 2019, doi: 10.1109/FIE43999.2019.9028612.
[33] M. Y. P. Peng, Y. Feng, X. Zhao, and W. L. Chong, "Use of Knowledge Transfer Theory to Improve Learning Outcomes of Cognitive and Non-cognitive Skills of University Students: Evidence From Taiwan," Front Psychol, vol. 12, Mar. 2021, doi: 10.3389/fpsyg.2021.583722.
[34] O. Chen, E. Retnowati, B. B. K. Y. Chan, and S. Kalyuga, "The effect of worked examples on learning solution steps and knowledge transfer," Educ Psychol (Lond), vol. 43, no. 8, pp. 914–928, 2023, doi: 10.1080/01443410.2023.2273762.
[35] B. B. Morrison, B. Dorn, and M. Guzdial, "Measuring cognitive load in introductory CS," pp. 131–138, 2014, doi: 10.1145/2632320.2632348.
[36] A. Zavgorodniaia, R. Duran, A. Hellas, O. Seppala, and J. Sorva, "Measuring the Cognitive Load of Learning to Program: A Replication Study," pp. 3–9, 2020, doi: 10.1145/3416465.3416468.
[37] L. Zhang, S. Kalyuga, C. Lee, and C. Lei, "Effectiveness of collaborative learning of computer programming under different learning group formations according to students' prior knowledge: A cognitive load perspective.," 2016, Assn for the Advancement of Computing in Education, Zhang, Liming: lmzhang@umac.mo.

Authors' Profiles

Carlos Sandoval-Medina is a doctoral candidate in the Information Systems Department at the Autonomous University of Aguascalientes, México. He is currently a full-time PhD student, a scholarship recipient of the CONAHCYT postgraduate program. His PhD thesis involves improving learning in introductory programming courses by applying the effects of cognitive load theory through a proposed roadmap model, addressing student motivational issues.

Estela L. Muñoz-Andrade obtained her Ph.D. in Exact Sciences and Information Systems from the Autonomous University of Aguascalientes (UAA) in 2010. She is currently a full-time Research Professor at the Department of Electronic Systems, UAA. She has been teaching for 25 years in the areas of programming fundamentals and data structures. Her area of research is the development of educational applications for learning programming. She is a member of the 'EGEL+ D-I COMPU' of the Academic Committee at CENEVAL, México, where she was the coordinator of the Programming Languages Academy from 2004 to 2017. She was the Head of the Electronic Systems Department from 2011 to 2017. Currently, she is Secretary of Undergraduate Teaching of the Basic Sciences Center of UAA.

Carlos A. Arévalo-Mercado received the Ph.D. degree in exact sciences and information systems from the Autonomous University of Aguascalientes (UAA), Mexico in 2010. He was the Head of the Information Systems Department, UAA, from 2014 to 2020, where he is currently a full-time Research Professor. His research interests include the design and development of learning technologies and teaching methods based on learning theories from cognitive sciences, particularly in introductory programming. He is a frequent collaborator with European universities, on multinational projects with Erasmus+ funding. He is a member of the National System of Researchers (SNII) of CONAHCYT, Mexico.

Jaime Muñoz-Arteaga is a full-time professor at UAA (Universidad Autónoma de Aguascalientes) in Aguascalientes, México. He holds a PhD in Computer Science from the University of Toulouse, France. He oversees the Computer Science area of the PhD in Applied Sciences and Technology. He has experience as leader of several research projects related to the digital divide, human-centered design for interactive systems and inclusive education. His research topics are in human-computer interaction, e-learning and artificial intelligence. He has published several books, two on software engineering, two on human-computer interaction and two on learning object technology.

How to cite this paper: Carlos Sandoval-Medina, Estela L. Muñoz-Andrade, Carlos A. Arévalo-Mercado, Jaime Muñoz-Arteaga, "Eliciting Knowledge Transfer and Self-management Skill through the Effects of Cognitive Load Theory on Programming Learning", International Journal of Modern Education and Computer Science(IJMECS), Vol.17, No.6, pp. 1-17, 2025. DOI:10.5815/ijmecs.2025.06.01