Available online at: http://journal.uny.ac.id/index.php/jpip

Jurnal Penelitian Ilmu Pendidikan

Volume 19, Nomor 1, 37-50, 2026

Didactic Design Implementation on Algorithms and Programming to Improve Students' High School Computational Thinking

Hassan Hidayatullah[1]*, Eka Fitrajaya Rahman[2], and Harsa Wara Prabawa[3]
[123]Computer Science Education, Universitas Pendidikan Indonesia, Indonesia
**Corresponding Author. e-mail*: hassanhid@upi.edu

Abstract
This research is motivated by the urgency of mastering 21st-century skills, particularly Computational Thinking (CT) in Informatics learning, and the prevalence of learning obstacles experienced by students. This study aims to develop a valid and effective didactical design to overcome students' learning obstacles and enhance their CT skills. The research method used is Didactical Design Research (DDR), which encompasses three stages: prospective analysis, metapedidactical analysis, and retrospective analysis. The research subjects involved Senior High School students in SMAN 9 Bandung. Data collection instruments included CT ability tests, interview guides, and classroom observations to identify ontogenic, didactical, and epistemological obstacles. The results indicate that the developed Hypothetical Didactical Design (DDH) proved effective, with a success rate of approximately 77.14% in overcoming students' learning obstacles. The implementation of this design also positively impacted the improvement of CT pillars, including decomposition, pattern recognition, abstraction, and algorithmic thinking. Furthermore, through retrospective analysis, the Empirical Didactical Design (DDE) was formulated to refine the design and address shortcomings identified in the DDH. This study recommends the application of learning obstacles-based didactical design as a responsive alternative strategy for Informatics learning.
Keywords: *Computational Thinking; didactical design; learning obstacles; informatics learning*

How to Cite (APA): Hidayatullah et al. 2026. *Didactic Design Implementation on Algorithms and Programming to Improve Students' High School Computational Thinking, 19*(2), 39–50. https://doi.org/10.21831/jpip.v19i1.94345

Received 10 Maret 2026; Received in revised from 17 April 2026; Accepted 04 Mei 2026

INTRODUCTION

The dynamics of education in the era of the 4.0 industrial revolution and society 5.0 demand curriculum adaptation that is responsive to the needs of 21st-century skills. Various global studies affirm that modern education must culminate in the mastery of digital literacy, complex problem-solving, and collaboration (Ekasari et al., 2025). In this context, Computational Thinking (CT) has been widely recognized as a fundamental competency, not only for computer scientists but as a basic life skill for every individual (Grover & Pea, 2018; Wing, 2006). The integration of CT into the high school curriculum aims to train students to think logically, algorithmically, and systematically in facing problems (Weintrop et al., 2016).

However, the transition toward Computational Thinking (CT)-based education in Indonesia still faces multidimensional challenges that have not yet been fully addressed in classroom learning practices (Mitrayana & Nurlaelah, 2023; Richardo et al., 2025). Field observations and preliminary studies indicate that students often struggle to understand abstract programming concepts, such as variables, branching logic, and loops (Herlina et al., 2023). These difficulties

are not only technical but also conceptual, especially in transforming everyday language logic into the rigid formal structure of programming languages (Kadar et al., 2021).

In the perspective of didactic situation theory, these obstacles can be categorized as learning obstacles that encompass epistemological, ontogenetic, and didactic aspects (Brousseau et al., 2001). Although various studies have identified the types and characteristics of these obstacles, most still stop at the stage of mapping student difficulties, without following up with the design of learning plans that systematically anticipate and reduce these obstacles in the context of real learning. On the other hand, research related to the development of CT learning tends to focus on improving learning outcomes or computational thinking skills in general, without explicitly linking it to the analysis of learning obstacles as the basis for designing didactic interventions. This indicates a gap between the diagnostic study of learning obstacles and the development of learning designs based on those obstacles.

Furthermore, there has not been much research that integrates the analysis of learning obstacles into the Didactical Design Research (DDR) framework to produce didactic designs that are not only theoretically valid but also empirically tested in overcoming learning obstacles while enhancing students' CT skills. Therefore, this study focuses on the development and implementation of DDR-based didactic designs specifically designed to anticipate learning obstacles in algorithm and programming materials, thereby bridging the gap between the analysis of learning obstacles and the practice of CT learning in the classroom. This problem is exacerbated by the learning approach that often jumps directly into coding activities without building a strong conceptual foundation. As a result, students are prone to experiencing misconceptions and cognitive overload (Bell et al., 2009). Therefore, a planned didactic intervention is needed to bridge that gap. This research offers a solution thru the development of a didactic design that integrates the ICARE model (Introduction, Connection, Application, Reflection, Extension) and the Computer Science Unplugged (CSU) approach. The CSU approach has proven effective in simplifying abstract concepts thru kinesthetic activities without computers (Threekunprapa et al., 2020), while the ICARE model provides a systematic learning structure (Hoffman & Ritchie, 1998).

Unlike previous research that focused more on final learning outcomes, this study uses the Didactical Design Research (DDR) framework to deeply explore how learning obstacles arise and how instructional design can minimize them (Suryadi, 2013). The main objective of this research is to produce a valid empirical didactic design to enhance students' computational thinking skills while reducing learning barriers in the subjects of algorithms and programming.

METHODS

This research uses a qualitative approach with the Didactical Design Research (DDR) method developed by Suryadi. This method was chosen due to its relevance in developing learning designs based on the analysis of learning obstacles and student learning trajectories. The research procedure is carried out thru three main stages: (1) prospective analysis, (2) meta pedagogical analysis, and (3) retrospective analysis.

Specifically, the implementation of DDR in this research focuses on the analysis of the didactical triangle, which includes the interactive relationship between the teacher, students, and learning materials. The focus of DDR is to create an optimal didactic situation by predicting and anticipating student responses (Kansanen, 2003; Suryadi, 2013). In designing learning, this research analyzes three fundamental relationships in the didactic triangle as illustrated in Figure 1.

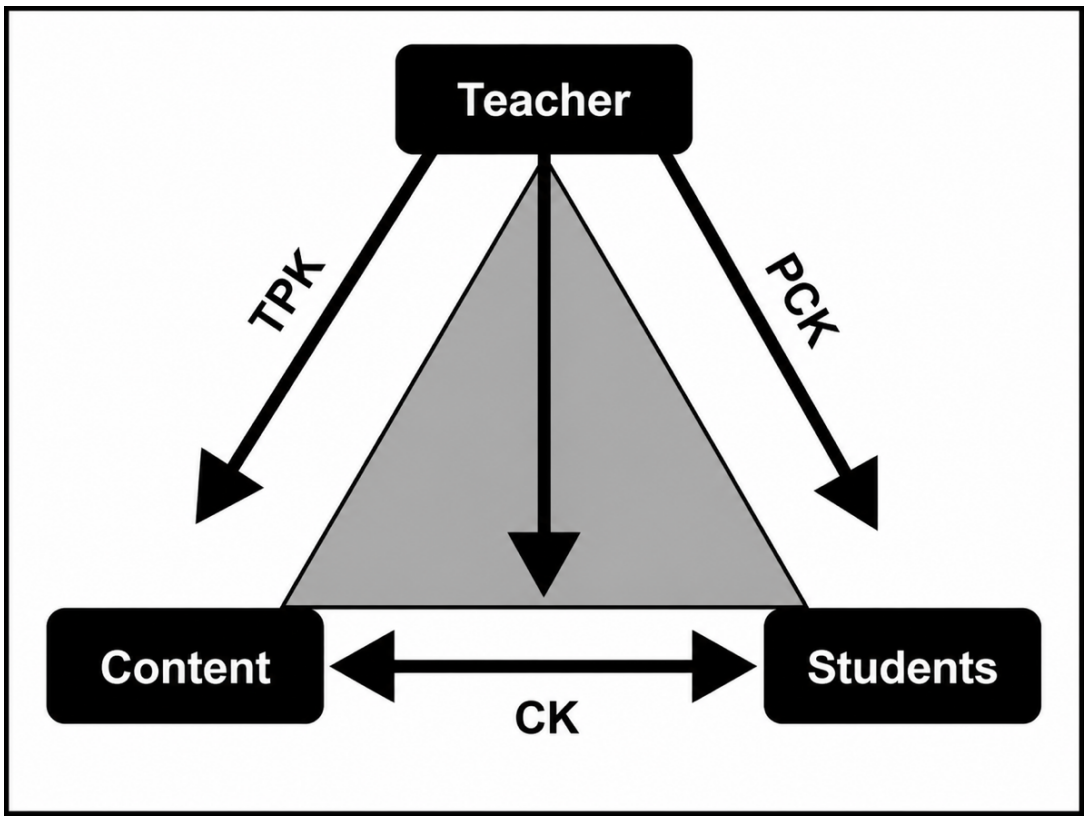

Figure 1. Didactic Triangle and the Relationship Between Components Adapted from (Kansanen, 2003; Suryadi, 2013)

Figure 1 shows that learning obstacles (LO) occur on the didactic relationship path (Student-Material). To address this, teachers must strengthen the Anticipatory Didactic Pedagogical (ADP) pathway by designing responses to predicted student difficulties (Suryadi, 2019). The didactic design in this study was developed thru three stages of DDR: The prospective analysis stage, which involves identifying initial Learning Objectives (LO) thru diagnostic tests and interviews to formulate a Hypothetical Didactic Design (HDD). The meta pedagogical analysis stage, which involves testing the HDD and observing the gap between the Hypothetical Learning Trajectory (HLT) and the Actual Learning Trajectory (ALT). The retrospective analysis stage, which involves revising the design based on field findings into an Empirical Didactic Design (EDD).

The research subjects are 36 eleventh-grade students at SMAN 9 Bandung. The research instruments include a computational thinking ability test (referring to the Brennan and Resnick (2012), interview guidelines, and observation sheets. Data analysis was conducted qualitatively to map the types of learning obstacles and the effectiveness of the design. This analysis aims to create didactic and pedagogical situations conducive to the development of Computational Thinking skills. In the preparation of the didactic design, the researcher not only designs the materials and teaching aids but also compiles the Student Response Prediction (SRP) and Didactic Pedagogical Anticipation (DPA) obtained from the responses to questionnaires and interviews with students who have previously engaged in algorithm and programming learning, or can be referred to as the phenomenon subjects. This is done so that the developed design is adaptive and responsive to various potential learning obstacles that may arise during the learning process. The DDR research flow used is illustrated in Figure 2.

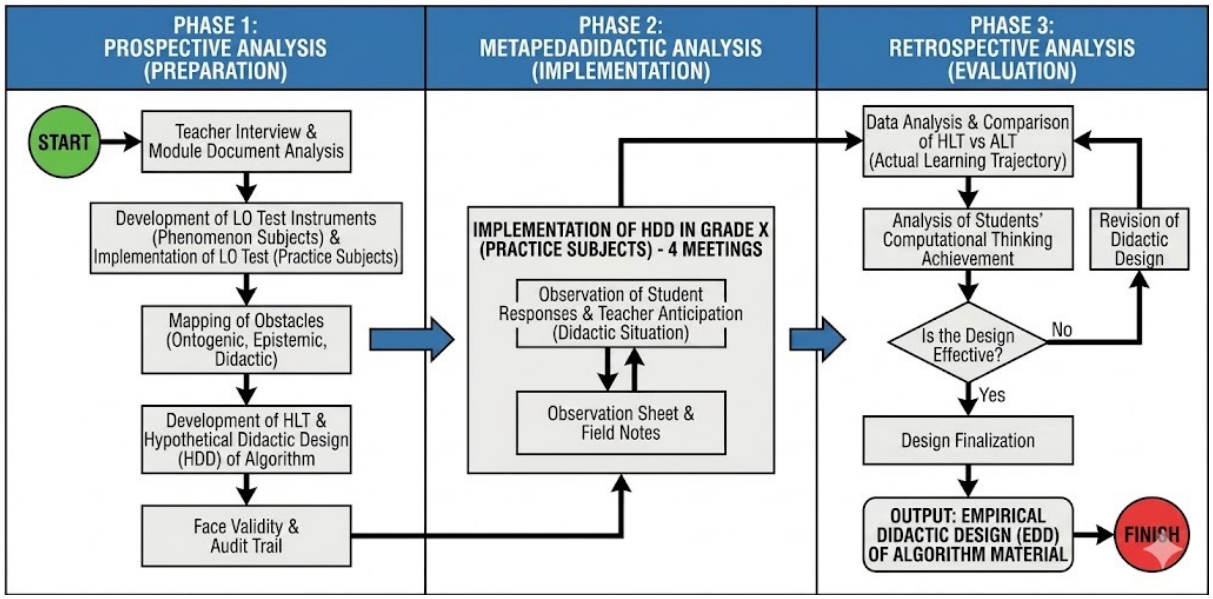

Figure 2. DDR research flow

At the stage of didactic or prospective analysis, researchers identify learning obstacles thru diagnostic tests and interviews to design the Hypothetical Didactic Design (HDD). The meta pedagogical analysis stage involves the implementation of the HDD in the classroom, where researchers observe the interactions between teachers, students, and learning materials. The final stage, retrospective analysis, is conducted by evaluating the effectiveness of the tested design and revising it into an Empirical Didactic Design (DDE). Details of the data collection techniques at each stage are presented in Table 1.

Tabel 1. Research Data Collection

Techniques Research	Stage Focus of Activities	Data Collection Instruments
Prospective Analysis	Identifying students' initial learning obstacles (LO) and learning trajectory.	Learning obstacles (LO) Ability Test, Interview Guidelines, Documentation Study.
Metapedagogical Analysis	Implementing Hypothetical Didactic Design (HDD) and observing student responses.	CT Ability Test, Learning Observation Sheet, Field Notes.
Retrospective Analysis	Evaluating the effectiveness of the design and revising it into Empirical Didactic Design (EDD).	Reflection Interview Guidelines, Data Triangulation.

The subjects of this research are the 11th-grade students of class 4 at SMAN 9 Kota Bandung, who were selected using purposive sampling technique. Data were collected using triangulation techniques, which included Computational Thinking ability tests, participatory observations during learning, and in-depth interviews. The test instruments were developed based on CT indicators (decomposition, pattern recognition, abstraction, algorithm) and have been validated by experts. Data analysis was conducted qualitatively descriptively to map the types of learning obstacles and the effectiveness of the developed didactic design.

RESULTS AND DISCUSSION

Results

The results of this research will focus on aspects that become the subject of analysis according to the stages of Didactical Design Research (DDR). The findings of this research will be outlined in several points as follows: (1) Identification of Learning Obstacles, (2) Hypothetical Didactic Design (HDD), (3) Effectiveness of Hypothetical Didactic Design, and (4) Empirical Didactic Design (EDD). The prospective analysis stage serves as the introduction to the research, where it will explain the obstacles encountered in algorithm and programming materials, the design of learning plans based on the findings of learning obstacles, up to the implementation of the

learning. The meta pedagogical stage will discuss the analysis of the findings from the implementation of DDH, particularly the relationship between the planned didactic situations between teachers and students and the material previously designed thru DDH with the actual implementation results, including the analysis of didactic anticipation of hypotheses (ADP) and the effectiveness of DDH. The retrospective analysis stage explains how the results of the previous meta pedagogical analysis can create a new didactic design, namely the empirical didactic design.

Identification of Learning Obstacles The initial stage of the research aims to uncover the learning obstacles experienced by students in the subject of Algorithms and Programming in Informatics related to Computational Thinking. Based on the results of the Learning Obstacles Test in the form of respondent ability tests (TKR) and interviews, it was found that students experienced significant difficulties when transforming story problems into algorithmic form. In this identification, learning obstacles will be divided based on sub-materials of algorithms and programming, including: (1) Pseudocode & Flowchart, (2) Data Types and Variables, (3) Branching Algorithms, and (4) Looping Algorithms. Based on the results of the Learning Obstacles Test, 35 specific LO were found spread across four main sub-materials. A summary of the learning obstacles findings is presented in Table 2, which refers to the recap of the field findings.

Table 2. Recap of Learning Obstacles (LO) Findings Based on Sub-Materials

Learning Material	Identified LO Code	Number of Findings	Description of Dominant Obstacles (Representative)
Flowcharts & Pseudocode	**LOFP1–LOFP7**	**7**	Students experienced misconceptions in the use of flowchart symbols (confusing input/output symbols with process symbols) and had difficulty translating natural language into pseudocode logic.
Data Types and Variables	**LOTDV1–LOTDV7**	**7**	(1) Epistemological Obstacle: Students failed to distinguish between Integer and String data types for numbers that are not used in operations (e.g., phone numbers considered as Integers). (2) Didactical Obstacle: Errors in writing variable declaration syntax due to confusion between the variable name and the data content ("container" vs. "content").
Branching Algorithms	**LOPC1–LOPC11**	**11**	Students tended to make jumping conclusions when determining TRUE/FALSE logical conditions and often ignored the implied ELSE condition (alternative) in word problems.
Looping Algorithms	**LOPU1–LOPU10**	**10**	The greatest difficulty lay in determining the termination condition (when the loop stops), leading to infinite loop logic or loops that were never executed at all.

Table 2 shows that although the obstacles are evenly distributed, the obstacles in the Data Types and Variables material (LOTDV1 to LOTDV7) receive special attention because they are the foundation for the subsequent material. Conceptual errors at this stage (for example, assuming all numbers can be added) have proven to hinder students' understanding of the looping material (counter loop). Empirical evidence of several students' answers that represent the aboveobstacles is presented in Figure 3.

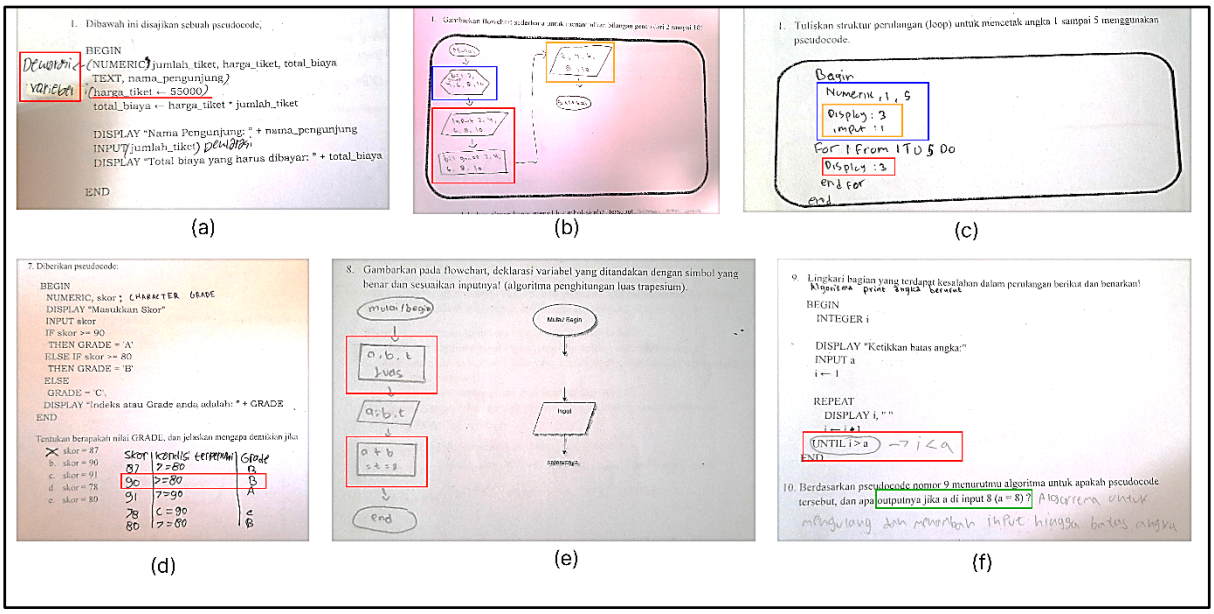

Figure 3. Example of Identification of Students' Epistemological Barriers in Various Sub-Materials

Analysis of Figure 3 shows specific error patterns: Algorithm Representation (Gb 2a and 2e) can be detailed in figure a where students do not understand the meaning of declaration, initialization, and variables, as well as in figure e where students experience misconceptions in the use of flowchart symbols, tending to use the same geometric shapes for different functions (input vs process) Data Types (Gb 2b) found students' inability to use variables, write algorithm notations, and differentiate between numeric and text data types, resulting in logical compilation failures (syntax errors) in their minds. Branching (Gb 2d) with students experiencing jumping to conclusions when determining true/false logical conditions, often neglecting the implicit else condition, resulting in incorrect output that should have been produced. Loops (Gb 2c and 2f) with obstacles in variable usage, the biggest obstacle is seen in iteration logic, where students struggle to determine the initial and final boundaries of the loop, resulting in infinite loop logic or non-functioning loops. These field findings confirm that students' learning obstacles are hierarchical; failures in understanding data types directly impact the failure to construct complex loop structures, as well as other subjects.

Hypothetical Didactic Design (HDD) To address these learning obstacles, the researcher developed a Hypothetical Didactic Design (HDD) implemented in four learning sessions. Each session is specifically designed to address each sub-topic that is the source of the obstacles, namely: (1) algorithm representation, (2) data types and variables, (3) branching algorithms, and (4) looping algorithms. The structure of this learning design was developed using the ICARE model (Introduction, Connection, Application, Reflection, Extension) combined with the CSU (Computer Science Unplugged) approach. This approach was chosen as a strategic solution to bridge the gap in student understanding by gradually transitioning the material from the concrete to the abstract stage. The CSU approach was strategically selected to address the abstraction barrier in the Data Type material (LOTDV-01 to LOTDV7) by incorporating the concept of "variables" into concrete physical activities before moving on to coding. Below is an example of the application of this model in the second implementation meeting, specifically on the topic of Data Types and Variables.

Table 3. Implementation of ICARE in Meeting DDH 2

Meeting	Introduction	Connection	Application	Reflection	Extension
Data Types and Variables Material	Introducing the topic of Data Types and Variables.	The relationship between flowcharts, pseudocode, and	Students engage in a simple game (data cards and data type boxes) to classify	Reflecting on and reviewing the learning outcomes.	Practice exercises and a brief introduction to programming languages

	data types & variables.	data according to its type.	and the implementation of data type variables.

At the Application stage, the development of learning materials for the implementation of CSU (Computer Science Unplugged) on data types and variables, the researchers adopted the CSU design pattern by Nishida et al. (2009) where this pattern is based on their analysis of successful CSU implementations. Details of the CSU implementation for this material will be explained in the table below.

Table 4. Computer Science Unplugged Pattern for Data Types and Variables Material

Data Types & Variables Material	
Problem	Data Types & Variables Problem: Students mistakenly understand variables as names; do not differentiate between declaration and initialization; confused about data types; do not understand data containers (LOTDV1–LOTDV7).
Context	Students' previous understanding is still based on language intuition; they do not yet have a mental model of data storage in memory; common mistakes related to variable naming rules.
Forces	a. High abstraction (the concept of a "container" is not physically visible). b. Learning habits that emphasize grading. c. Limited examples of visual representations of memory. d. Epistemological learning objectives related to variable structure & data types.
Solution	CSU activity, using the analogy of container and content with glasses/boxes as variables to explain the concept of data types and variables. The teacher explains that a container can only be filled with one piece of data, and that data must match the type of the container/variable. The teacher and students discuss to notate it on the whiteboard by writing the variable pseudocode. Below is an example of the media used:  Sorting Game with number/letter/text/boolean cards using cards and containers or boxes to validate students' knowledge formulation. The teacher distributes cards to several students and asks them to categorize the data according to data types using boxes (to be placed inside). The teacher can adjust the media requirements; for example, if they want to conduct a step-by-step understanding, they can first categorize the data based on numbers and letters, and then ask and validate each type of data that has been collected in the box one by one. Description of the media used is as follows:  Socratic questioning to guide the classification of data types & value changes or swap algorithms. Visualization of declaration and initialization notation on the whiteboard.

Resulting Context	Students realize that a variable is a "container" that holds a single value and can change; understand the types of data that can be entered; write declarations and initializations correctly.
Rasionale	Concrete activities help overcome abstraction barriers and strengthen the ability to abstract data and recognize patterns in computational thinking.

In this framework, a Hypothetical Learning Trajectory (HLT) is developed as an anticipation of students' thinking processes. In its design, the researcher used the Theory of Didactical Situation (TDS) as a paradigm in building knowledge epistemically (Epistemology) (Suryadi, 2019). As a representation, Table 5 presents an example of HLT designed for the meeting on Data Types and Variables, where the learning design focuses on addressing students' confusion in the definition and classification of data.

Table 5. Hypothetical Learning Trajectory (HLT) Data Types and Variables Meeting

No.	Learning Objectives (Sub-Objectives)	Learning Activities (Teacher/Student Activities)	Hypothesis of the Learning Process (Expected Response, LO, and Didactic Anticipation)
1.	Students understand the basic concept of a variable as a container for storing values in an algorithm.	The teacher uses the CSU activity "Container and Content" analogy: students hold a glass/box as a variable and place an object (value) inside it. Socratic Questioning Discussion: "Can one container hold two contents?"	Some students might still think of variables as just random names or symbols (LOTDV1, LOTDV2). The teacher emphasizes the concept that "a variable stores a value that can change," and connects it with programming logic.
2.	Students learn about different types of data (integer, float, character, string, boolean) and their characteristics.	The teacher distributes cards with various values (25, 3.14, 'A', "Kevin", True/False) and then invites students to categorize the cards into data type categories (CSU Sorting Game).	Students tend to misclassify (for example, thinking 3.14 = integer, or "A" is the same as 'A') because they do not yet understand data representation (LOTDV3, LOTDV4). The teacher guides thru comparison questions and real programming examples.
3.	Students understand the rules for writing variable names and their relationship with data types.	The teacher writes examples on the board: 1name, name student, name_student, @age, total_price. Students discuss which ones are valid and the reasons.	Students often focus on the results without understanding the syntax rules (LOTDV5). The teacher uses direct validation techniques (writing valid-invalid examples) so that students understand the reasoning behind variable naming rules.
4.	Students are able to distinguish between variable declaration and initialization..	The teacher shows two simple lines of code: int x; and x = 5; then asks, "What is the difference between these two statements?" The discussion is directed so that students discover the concepts of declaration and initialization	Potential LO: students consider both the same or interchange their order (LOTDV6). The teacher emphasizes the difference in function and demonstrates the consequences of incorrect order in code
5.	Students can relate the concept of data types and variables to the context of simple algorithms or pseudocode.	The teacher gives a case study: "Create an algorithm to calculate the area of a rectangle." Students determine the necessary variables and their data types.	Students may not yet be able to relate variables to values in an algorithmic context (LOTDV7). The teacher guides with leading questions: "Which value changes?" Which variable holds the result?" and conclude with a reflection on the relationship between data type–variable–algorithm.

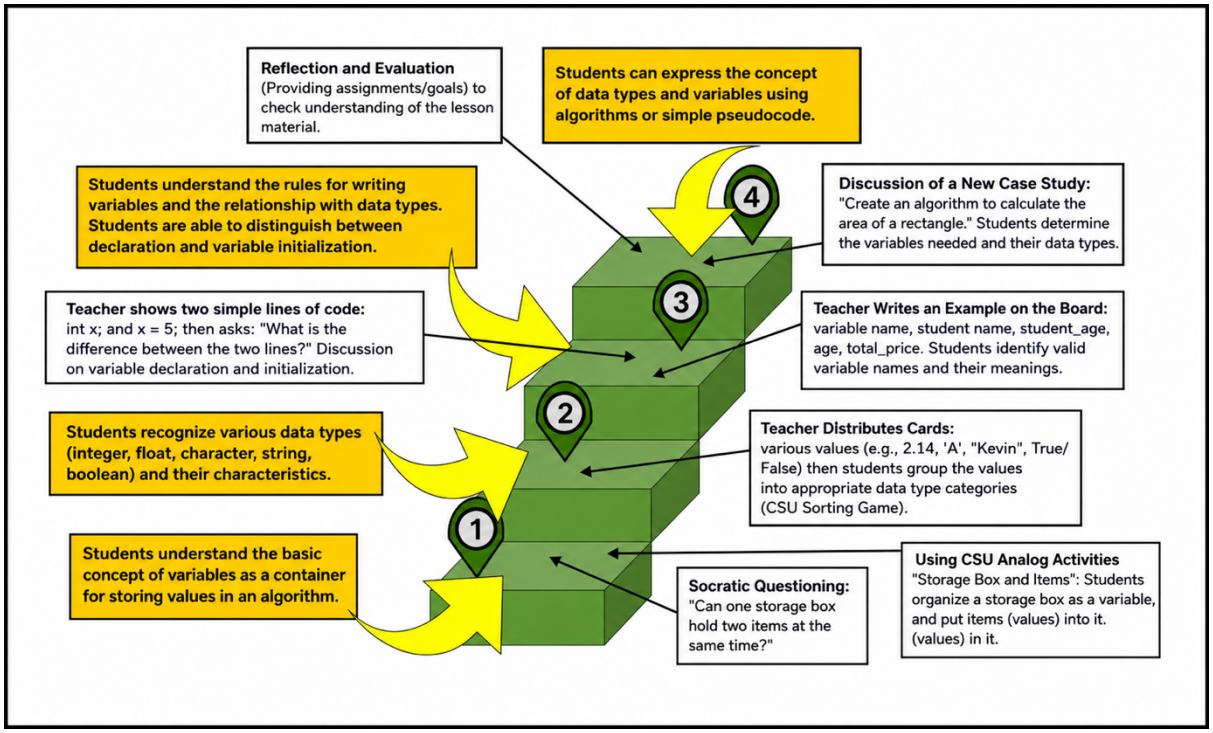

Figure 4. Hypothetical Learning Trajectory (HLT) Meeting on Data Types and Variables

In this research, 4 HLTs were created based on the previously mentioned sub-materials. From the 4 HLTs created, 4 DDHs will be derived, further divided by sub-topics or sub-designs. As an example, based on the HLT above, the DDH is developed by integrating Student Response Prediction (PRS) and Pedagogical Didactic Anticipation (PDA). An example of one implementation of the DDH for meeting 2 on the sub-material Data Types and Variables with the sub-topics of the definition of data types and variables and types of data will be presented in the table below.

Table 6. Hypothetical Didactic Design of the sub-material Data Types and Variables

No.	Sub Topic (Sub Design)	Predicted Student Response	Didactic and Pedagogical Anticipation
1.	Definition of Data Types and Variables	Students understand variables only as symbolic names without relating them to the function of data storage. Students try to place more than one data in one variable because they do not yet understand the limitation of a single value. Students consider data types and variables to be the same thing.	Action Situation: The initial stimulus used CSU activities and Socratic questioning. The teacher reminded the previous lesson using the analogy of a container (bowl) and data (contents). The teacher prepared a bowl with a whiteboard sticker so it could be written on with a marker, and several cards with paper attached to them, on which example data (such as 17, 3.14, 'Hadi') were written. After discussing the variable names (such as age, balance, name) that should be written on the sticker, the teacher demonstrated the process of inserting a data card. Key questions using Socratic questioning include: a. What does the temporary notation mean when there is only this bowl as a container for the variable name before the data is entered? b. If this card is the data and we put it into the variable, what is the notation now? c. If I put another card into this bowl, can it still be done? what would the notation be like? d. What happens to the previous data? see the following notation!

			The teacher writes on the board: variable_name = 0 variable_name = data, then tries to write variable_name = data1 = data2 a. Do you think this method can be used? b. Is this notation correct and readable? Formulation Situation: Students are asked to write the definition of a variable in their own words based on the previous simulation. Validation Situation: Validation related to the introduction of this concept is conducted thru a whole-class discussion followed by teacher intervention if deemed necessary. At a minimum, students understand the key concept that a variable is a container for storing a single data value at a time, and a data type is the kind of value that can be stored within it.
2.	Types of Data Types	a. Students do not yet understand that data types affect how computers read and process data. b. Students mix numbers, letters, and text as the same type of data. c. Students find it difficult to categorize data types because they only look at the external form, not the nature of the value. d. Some students might be confused about distinguishing between character data and strings ('A' and "Kevin" are both letters, right?). e. Students actively try to categorize cards quickly without thinking of a logical reason.	Action Situation: Continue using the CSU Activity and Socratic questioning. The teacher introduces four types of data (integer, float, character, string). The teacher prepared cards with various data written on them (such as: 25, 3.14, 'A', "Kevin", "RC", and others) and distributed them randomly to all the students. The teacher prepared two boxes labeled "NUMBERS" and "LETTERS" and guided the students to place the distributed data into the boxes according to their types. The Socratic questioning used to guide is: Please put all the cards into the box that you think is the appropriate type, why did you put this data here? (Discuss each piece of data) a. If your card has the number 25 written on it, which box would you put it in? b. If it says 'A', is it a number or a letter? c. Now let's classify the data types of each piece of data you put in the box. d. What pattern distinguishes between integer and float data? e. Why is the data 'A' classified as a character and not a string? f. What distinguishes a character from a string then? g. What about the data "RC"? Does it also include characters? Next, the teacher added that there is data that can only be filled with true and false, which is the boolean data type. Formulation Situation: Students discuss the reasons they chose certain boxes, then the teacher writes the data type categories on the board. Validation Situation: The teacher emphasizes the characteristics of each data type based on the classification results of the students and demonstrates examples in the form of simple program writing (for example, Python, C, or pseudocode).

The gap between HLT and ALT (Metapedagogical Analysis)

The implementation of DDH in the classroom shows an interesting dynamic of interaction between the planned Hypothetical Learning Trajectory (HLT) and the reality of the students' learning process (Actual Learning Trajectory/ALT). In-depth comparisons were conducted specifically on the topic of Data Types and Variables (Meeting 2), where many learning obstacles related to the understanding of abstract concepts were found. From this analysis, we can outline the effectiveness of the DDH that has been implemented. Table 7 presents a comparison between HLT and ALT on three key activities in the meeting. This data is summarized from field observations during the design implementation.

Table 7. Comparison of HLT and ALT

Learning Activities	HLT	ALT	Gap Analysis
Aktivitas 1: Analogi Wadah (Pengenalan Konsep Variabel)	Container Analogy (Introduction to the Concept of Variables) Students are predicted to understand the concept of variables as "containers" for storing values thru the analogy of containers with data cards.	Students respond positively and quickly grasp the concept of containers. They are able to state that "different containers for different contents."	As Expected. Concrete analogies effectively bridge students' initial understanding of the abstract concept of variables.
Activity 2: Data Classification (Understanding Data Types)	Students are expected to accurately distinguish between Integer, Float, Character, String, and Boolean data. Including distinguishing between countable numbers and numbers as labels, as well as text as characters or strings.	Most students successfully grouped the data. However, there was confusion with the data of one letter like 'A'. The students thot it was a String because it looked like a letter, but it should have been a character since it was only one letter and did not use double quotes.	Partially Achieved. An epistemic barrier was found where students were trapped in the visual representation of numbers without understanding their logical function.

Revision Toward Empirical Didactic Design (Retrospective Analysis)

Based on the analysis of the gap between HLT and ALT, as well as the discovery of new learning obstacles that emerged during implementation, a retrospective revision was conducted to produce the Empirical Didactic Design (DDE). DDE is an enhancement of the DDH that has been empirically tested in the field. The revision focuses on aspects that still show gaps in student understanding, particularly in the transition from concrete to abstract concepts. The details of the revision are presented in Table 8.

Table 8. Revision Toward Empirical Didactic Design (DDE)

Design Component	Weaknesses in DDH (Field Findings)	Improvements in DDE (Revision)
CSU Activity	Students tend to focus on the physical competition aspect in the CSU simulation and forget its algorithmic logic.	Verbalization Protocol: Adding a "Stop & Think" rule where students must verbalize the logical reasoning (e.g., "Because the value of X < Y, then...") before performing physical actions in the simulation.
Learning Media	The use of text-based slides and worksheets sometimes confuses students who experience epistemological barriers in understanding textual instructions for problem decomposition.	Addition of Infographics & Visuals: Slides and worksheets are equipped with visual icons and simple flow diagrams to help students visualize the steps of decomposition and reduce cognitive load.
Anticipation Scenario (ADP)	Teacher anticipation that is only verbal (Socratic questions) is sometimes less effective for passive or slow-responding students.	Provision of "Hint Cards": Physical cards containing step-by-step instructions are provided, which students can take independently if they encounter a deadlock, fostering learning independence.
Material Structure:	The Data Types and Variables material is too dense in one session, causing confusion between the concepts of declaration vs initialization.	Restructuring Time: The material is divided into two focused sessions: (1) Concepts & Classification (Unplugged), and (2) Syntax & Operations (Plugged), providing cognitive breaks for students.

Discussion

Characteristics of Student Learning

Barriers The findings of this study indicate that student learning barriers in algorithm and programming material are complex and hierarchical, with epistemological barriers as the main root that influences the emergence of other barriers. Difficulty in abstracting from the context of the problem to a formal algorithm indicates a gap between students' intuitive knowledge and the symbolic representation required in programming. The phenomenon of jumping to conclusions shows that students do not yet have a systematic algorithmic thinking structure, leading them to be outcome-oriented without going thru a logical thinking process. This condition aligns with recent research findings that state the main difficulty in programming learning lies in the process of translating conceptual understanding into symbolic representation (Moors et al., 2018). Therefore, the obstacles that arise cannot be viewed merely as individual mistakes, but rather as

indicators that the learning design is not yet fully aligned with the way students construct knowledge.

The findings then became an important basis for designing more responsive learning interventions. By understanding that obstacles stem from an immature cognitive process, didactic design needs to be able to provide a bridge that connects students' concrete experiences with formal abstractions. This emphasizes that the effectiveness of learning is not only determined by the material presented but also by how the learning structure is designed to accommodate the students' thinking processes gradually. Effectiveness of ICARE and CS Unplugged-Based Design Based on the characteristics of these obstacles, the didactic design developed in this study demonstrates effectiveness as it can facilitate the cognitive transition from concrete to abstract. The integration of ICARE provides a systematic learning flow, while the Computer Science Unplugged approach serves as a conceptual bridge that lowers the level of abstraction in the early stages of learning. Activities such as the "container" analogy and data classification serve as cognitive anchors that help students build an initial understanding before interacting with symbols and syntax. These findings are in line with research showing that the unplugged approach is effective in reducing cognitive load and enhancing understanding of basic programming concepts (Brackmann et al., 2017; Threekunprapa et al., 2020).

However, the effectiveness is not entirely uniform. At the advanced stage, particularly when students begin to encounter symbolic representations, obstacles in the application of syntax and consistency of algorithmic logic are still found. This indicates that concrete experiences do not automatically lead to strong abstractions unless followed by a directed reflection process. Sentance et al. (2019) also emphasize that the transition from concrete activities to formal representations is a critical point in programming learning. Thus, the success of the design does not only depend on the use of concrete activities but also on the quality of teacher facilitation in guiding conceptual reflection.

The Role of Didactic Anticipation in Reducing Barriers These limitations then lead to the importance of the role of Pedagogical Didactic Anticipation (PDA) in strengthening the effectiveness of learning design. In this study, the Didactic Anticipation Pedagogical (ADP) was developed based on Student Response Prediction (PRS), so that the learning was designed to be anticipatory of potential difficulties that may arise. This approach allows teachers to provide more timely and contextual interventions, either thru guiding questions or adaptive scaffolding. These findings are in line with the research by Prediger, (2019), which shows that a prediction-based learning design can enhance the effectiveness of interventions and reduce misconceptions.

However, the implementation of ADP also presents challenges, especially in accommodating the diversity of student characteristics. Not all students respond to interventions in the same way, so flexibility in the application of didactic strategies is necessary. This reinforces the findings of Sentance et al. (2019) which emphasize the importance of differentiation in computational learning. Thus, ADP cannot be viewed as a single solution, but rather as a framework that needs to be dynamically adapted according to the classroom context. Improvement of Computational Thinking Skills Overall, the interaction between learning design, concrete activities, and didactic anticipation contributes to the enhancement of students' Computational Thinking skills. The changes that occurred are not only visible in the final results but also in the students' thinking processes, which have become more structured. Students are beginning to be able to decompose problems, recognize patterns, perform abstraction, and construct algorithms more systematically. These findings are consistent with the research of Grover (2017) and Taylor et al. (2025), which show that structured and contextualized learning can enhance the quality of students' computational thinking.

Compared to previous research, which generally separates the analysis of learning difficulties and the development of learning, this study offers a more integrative approach by making learning obstacles the main basis in the design of didactic design. Therefore, the novelty of this research lies in the integration of learning obstacle analysis, the DDR framework, as well

as the ICARE and Computer Science Unplugged approaches into a coherent and empirically based design. This approach not only provides practical contributions to Informatics education but also enriches the conceptual framework on how didactic design can be used to systematically and sustainably manage learning barriers.

CONCLUSION

This research emphasizes that students' learning obstacles in algorithm and programming material are systemic and interconnected, where difficulties in understanding abstract concepts cannot be separated from the aspect of learning design that is less responsive to students' cognitive characteristics. These findings indicate that efforts to enhance Computational Thinking skills are insufficiently achieved thru technical programming exercises alone but require a reconstruction of the learning approach based on the analysis of students' learning obstacles. The main contribution of this research lies in the integration of learning obstacles analysis into the Didactical Design Research (DDR) framework to produce didactic designs that are not only theory-based but also empirically validated in real learning contexts. The developed design shows that the combination of the conceptual approach (ICARE) and concrete experiences (Computer Science Unplugged) can serve as a transitional bridge from intuitive understanding to more formal algorithmic thinking. Thus, this research expands the practice of Informatics learning design by placing learning obstacles as the starting point for designing didactic interventions. The implication of these findings is the need for a paradigm shift in Informatics education, from being oriented toward mastering syntax to developing a gradual, contextual, and adaptive computational thinking structure. Teachers do not only play the role of delivering material but also as designers of didactic situations capable of anticipating various possible responses and difficulties of students. The direction of the follow-up research opens opportunities to test the applicability of the produced didactic design in a broader context, both in other Informatics subjects and at different educational levels. In addition, further research can explore the integration of this didactic design with digital technology or project-based learning environments to see the consistency of its effectiveness in sustainably enhancing Computational Thinking skills.

REFERENCES

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). *Computer Science Unplugged: school students doing real computing without computers*.

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017). Development of Computational Thinking Skills through Unplugged Activities in Primary School. *WiPSCE '17: 12th Workshop in Primary and Secondary Computing Education*, 65–72. https://doi.org/10.1145/3137065.3137069

Brennan, K., & Resnick, M. (2012). *New Frameworks for Studying and Assessing the Development of Computational Thinking BT - Proceedings of the American Educational Research Association Annual Meeting*. American Educational Research Association.

Brousseau, G., Brousseau, N., & Warfield, V. (2001). An experiment on the teaching of statistics and probability. *The Journal of Mathematical Behavior*, *20*(3), 363–411. https://doi.org/10.1016/S0732-3123(02)00078-0

Ekasari, L. A., Firdaus, F. M., & Pawestri, S. A. (2025). Analysis of the Profile of Basic Mathematical Skills and Misconceptions of Prospective Elementary School Teachers. *Jurnal Penelitian Ilmu Pendidikan*, *18*(2), 133–144. https://doi.org/10.21831/jpip.v18i2.93196

Grover, S. (2017). Assessing Algorithmic and Computational Thinking in K-12: Lessons from a Middle School Classroom. In P. J. Rich & C. B. Hodges (Eds.), *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 269–288). Springer International Publishing. https://doi.org/10.1007/978-3-319-52691-1_17

Grover, S., & Pea, R. (2018). *Computer Science Education: Perspectives on Teaching and*

Learning in School. Bloomsbury Academic. https://doi.org/10.5040/9781350057142

Herlina, N., Sesmiarni, Z., Zakir, S., & Ilmi, D. (2023). Analisis Hambatan Belajar Siswa Pada Mata Pelajaran Informatika di MTsN 6 Agam. *Journal of Educational Management and Strategy*, *2*(1), 86–103. https://doi.org/10.57255/jemast.v2i1.231

Hoffman, B., & Ritchie, D. (1998). Teaching and Learning Online: Tools, Templates, and Training. *SITE 98: Society for Information Technology & Teacher Education International Conference Proceedings*.

Kadar, R., Abdul Wahab, N., Othman, J., Shamsuddin, M., & Mahlan, S. B. (2021). A Study of Difficulties in Teaching and Learning Programming: A Systematic Literature Review. *International Journal of Academic Research in Progressive Education and Development*, *10*(3), Pages-591-605. https://doi.org/10.6007/IJARPED/v10-i3/11100

Kansanen, P. (2003). Studying--the Realistic Bridge Between Instruction and Learning. An Attempt to a Conceptual Whole of the Teaching-Studying- Learning Process. *Educational Studies*, *29*(2–3), 221–232. https://doi.org/10.1080/03055690303279

Mitrayana, M., & Nurlaelah, E. (2023). Computational Thinking in Mathematics Learning: Systematic Literature Review. *Indonesian Journal of Teaching in Science*, *3*(2), 133–142. https://doi.org/10.17509/ijotis.v3i2.60179

Moors, L., Luxton-Reilly, A., & Denny, P. (2018). Transitioning from Block-Based to Text-Based Programming Languages. *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, 57–64. https://doi.org/10.1109/LaTICE.2018.000-5

Nishida, T., Kanemune, S., Idosaka, Y., Namiki, M., Bell, T., & Kuno, Y. (2009). A CS unplugged design pattern. *SIGCSE09: The 40th ACM Technical Symposium on Computer Science Education*, 231–235. https://doi.org/10.1145/1508865.1508951

Prediger, S. (2019). Theorizing in Design Research. *Avances de Investigación En Educación Matemática*, *15*, 5–27. https://doi.org/10.35763/aiem.v0i15.265

Richardo, R., Dwiningrum, S. I. A., Murti, R. C., Wijaya, A., Adawiya, R., Ihwani, I. L., Ardiyaningrum, M., & Aryani, A. E. (2025). Computational thinking skills profile in solving mathematical problems based on computational thinking attitude. *Journal of Education and Learning (EduLearn)*, *19*(2), 1157–1166. https://doi.org/10.11591/edulearn.v19i2.21643

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, *29*(2–3), 136–176. https://doi.org/10.1080/08993408.2019.1608781

Suryadi, D. (2013). *Didactical Design Research (DDR): Strategi Mengembangkan Pembelajaran Matematika*. Rizqi Press.

Suryadi, D. (2019). *Landasan Filosofis Penelitian Desain Didaktis (DDR)* (Vol. 10). Pusat Pengembangan DDR Indonesia.

Taylor, R., Fakhimi, M., Ioannou, A., & Spanaki, K. (2025). Personalized learning in education: a machine learning and simulation approach. *Benchmarking: An International Journal*, *32*(7), 2662–2689. https://doi.org/10.1108/BIJ-06-2023-0380

Threekunprapa, A., Yasri, P., & Dr. Mahidol University, Thailand, pratchayapong.yas@mahidol.edu, I. for I. L. (2020). Unplugged Coding Using Flowblocks for Promoting Computational Thinking and Programming among Secondary School Students. *International Journal of Instruction*, *13*(3), 207–222. https://doi.org/10.29333/iji.2020.13314a

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215