Vol. 10, No. 01, January 2026, pp. 14-23

INTERNATIONAL JOURNAL OF INNOVATION IN ENTERPRISE SYSTEM

Journal homepage: https://ijies.sie.telkomuniversity.ac.id
e-ISSN: 2580-3050, DOI: https://doi.org/10.25124/ijies.v10i01.10530

Diagnosing Object-Oriented Programming Difficulties: Association Rule Mining and Block-Based Scaffolding

Andre Rangga Gintara[1], Lala Septem Riza[2], Wahyudin[3*]

[1] Department of Computer Science Education
Universitas Pendidikan Indonesia, Bandung, Indonesia
andrerangga823@upi.edu

[2] Department of Computer Science Education
Universitas Pendidikan Indonesia, Bandung, Indonesia
lala.s.riza@upi.edu

[3] Department of Computer Science Education
Universitas Pendidikan Indonesia, Bandung, Indonesia
wahyudin_sanusi@upi.edu

*Corresponding Author

ARTICLE INFO

Article history:
Received 12 January 2026
Accepted 18 February 2026
Published 17 July 2026

ABSTRACT IN ENGLISH

Logical thinking is fundamental for Object-Oriented Programming, yet vocational students frequently struggle with its abstract concepts. This study aims to overcome these cognitive barriers through a data-driven diagnostic approach and targeted intervention. Utilizing Educational Data Mining techniques, specifically Association Rule Mining, this research analyzed pretest patterns to map learning difficulties among 35 software engineering students in Indonesia. A Research and Development method with a One-Group Pretest-Posttest design was employed. The analysis revealed a "cognitive domino effect," identifying that failures in advanced topics are rooted in specific prerequisite weaknesses. Based on these findings, a remedial intervention using a Scaffolding model assisted by Block-Based Programming was implemented. The results demonstrated that this data-driven intervention significantly enhanced students' logical thinking skills ($p<0.001$), with a substantial increase in N-Gain scores. It is concluded that integrating algorithmic diagnosis with visual scaffolding effectively bridges the gap between abstract concepts and practical coding skills, offering a scalable model for vocational education.

Keywords:
Association Rule Mining;
Block-Based Programming;
Educational Data Mining;
Object-Oriented Programming; Scaffolding;

1. INTRODUCTION

Logical thinking and computational problem-solving are fundamental competencies required in the 21st-century workforce [1]. In the context of vocational education, these skills are predominantly developed through programming courses to meet industry demands. However, global assessments such as PISA 2022 indicate a stagnation in logical reasoning skills among students [2], which poses a significant challenge for educators in technical fields. This gap is particularly concerning in vocational high schools, where graduates are expected to be work-ready but often lack the fundamental logic required for complex software development.

Specifically, Object-Oriented Programming (OOP), which is a paradigm essential for modern software engineering, presents substantial cognitive barriers for novice students. Unlike procedural programming, OOP requires a shift in thinking to understand abstract concepts such as polymorphism, inheritance, and encapsulation [3]. These concepts are inherently abstract and difficult to visualize for beginners [14]. Consequently, students often struggle to translate these abstract concepts into concrete code, leading to a gap in practical programming skills and a high rate of learning failure [4], [5].

To address these difficulties effectively, educators must first move beyond intuition and identify the specific learning barriers students face using empirical data. Educational Data Mining (EDM) has emerged as a critical tool in this domain. According to Han et al., data mining techniques allow researchers to discover hidden patterns and correlations within large datasets [6]. Specifically, Association Rule Mining (ARM) is a technique that can reveal "if-then" patterns in student errors (e.g., "If a student fails in Encapsulation, they are 90% likely to fail in Inheritance"). This granular diagnosis is superior to simple score aggregation because it pinpoints the root cause of failure rather than just the symptoms [8].

However, a review of existing literature reveals a significant research gap. Most current studies utilize EDM and ARM primarily for predicting student dropout rates or classifying general academic performance [7], [9]. There is a scarcity of research that utilizes ARM for diagnostic purposes in specific subject matter, like OOP, to map the prerequisite misconceptions. Furthermore, few studies connect these diagnostic findings directly to a pedagogical intervention. Diagnosis without intervention is insufficient to solve the learning problem.

To bridge this gap, this study proposes an integrated solution using the Scaffolding method. Grounded in Vygotsky's Zone of Proximal Development (ZPD) theory, scaffolding suggests that students can master complex tasks if provided with structured support [1]. In the context of programming, Block-Based Programming environments (such as Scratch or Blockly) serve as effective technological scaffolds. Recent meta-analyses confirm that block-based environments significantly reduce cognitive load and improve academic achievement compared to text-based coding alone [10], [11].

The primary novelty of this research lies in its integrated "closed-loop" framework. While previous studies have utilized Educational Data Mining solely for prediction [7] or Scaffolding solely for teaching [12], this study bridges the two domains. It utilizes ARM not just to predict grades, but to diagnostically map specific prerequisite misconceptions, which then directly trigger a targeted Block-Based Scaffolding intervention. This ensures that the remedial process is not generic but data-driven and precise. The conceptual framework of this research is illustrated in Figure 1.

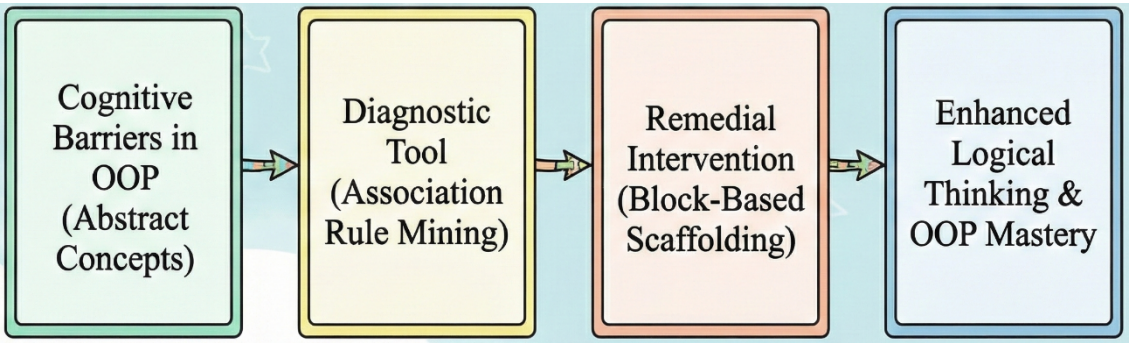


Figure 1 - Conceptual framework from diagnosis to scaffolding intervention

As shown in Figure 1, the framework addresses the cognitive barriers in OOP by first diagnosing specific error patterns using ARM. This data-driven diagnosis then triggers a targeted remedial intervention using Block-Based Programming as a scaffolding tool to reinforce understanding before returning to text-based OOP. Based on this framework, the objectives of this study are: (1) to detect student learning difficulty patterns in OOP using Association Rule Mining, and

*Corresponding author: wahyudin_sanusi@upi.edu

(2) to implement and evaluate a remedial learning flow using a Scaffolding model assisted by Block-Based Programming based on the diagnosis results.

2. METHOD

This study employed a mixed-methods design with a Sequential Explanatory model, integrating Research and Development (R&D) with Educational Data Mining (EDM). The R&D phase adopted the ADDIE model (Analysis, Design, Development, Implementation, Evaluation) to develop valid Block Programming learning media. Simultaneously, the EDM phase focused on extracting difficulty patterns from student assessment data to guide the implementation of the media [6].

2.1 Research Subjects and Instruments

The research subjects were 35 tenth-grade students from the Software Engineering program at a state vocational high school in Cimahi, Indonesia. The subjects were selected using purposive sampling with specific criteria: (a) students enrolled in the semester in which OOP was taught, and (b) students who had not previously received formal OOP instruction. This ensured that the pre-test data purely reflected their initial cognitive state and fundamental misconceptions.

The primary data collection instrument was a logical thinking ability test consisting of 60 multiple-choice items. The items were designed to comprehensively cover three core OOP topics: Class & Object, Encapsulation, and Inheritance. Each item was mapped to specific logical thinking aspects: Sequential Thinking, Argumentation Skill, and Conclusion Drawing. The instrument underwent expert validation by university lecturers specializing in computer science and vocational education to ensure content and construct validity [13], [19]. The mapping distribution is detailed in Table 1.

Table 1 - Mapping of Test Items to OOP Topics and Logical Aspects

Questions	Topic	Logical Thinking Aspect
1, 2, 21, 22, 41, 42, 59	Class & Object	Sequential Thinking
3, 4, 23, 24, 43, 44	Class & Object	Argumentation Skill
5, 6, 25, 26, 45, 46	Class & Object	Conclusion Drawing
7, 8, 27, 28, 39, 40, 47, 48	Encapsulation	Sequential Thinking
9, 10, 29, 30, 49, 50, 60	Encapsulation	Argumentation Skill
11, 12, 31, 32, 51, 52	Encapsulation	Conclusion Drawing
13, 14, 19, 33, 34, 53, 54	Inheritance	Sequential Thinking
15, 16, 20, 35, 36, 55, 56	Inheritance	Argumentation Skill
17, 18, 37, 38, 57, 58	Inheritance	Conclusion Drawing

2.2 Data Analysis Procedure

The data analysis procedure was systematically designed by adapting the Knowledge Discovery in Databases (KDD) framework [6]. The comprehensive workflow, ranging from the initial data collection to the discovery of learning difficulty patterns, is visualized in Figure 2.

*Corresponding author: wahyudin_sanusi@upi.edu

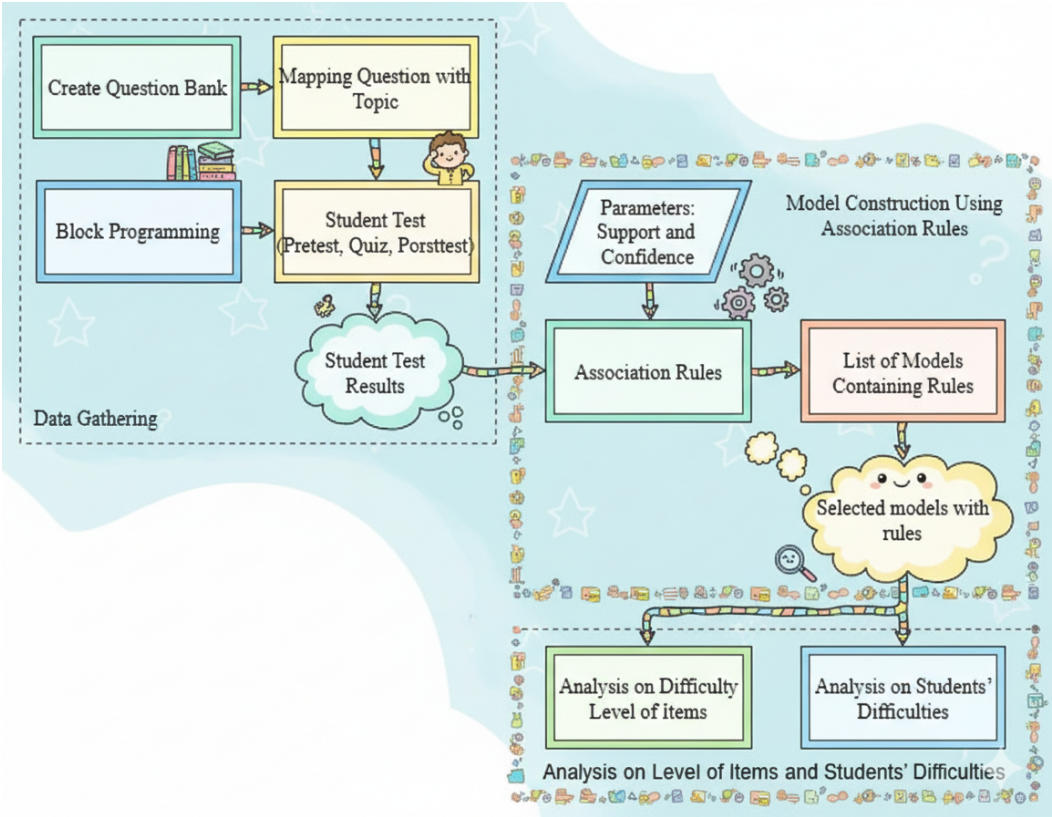


Figure 2 - Computational model of association rule mining for OOP difficulty detection

As illustrated in Figure 2, the analysis workflow consists of three sequential phases:

1. **Data Collection and Transformation:** Student pre-test data were collected and mapped to the predefined OOP topics. The raw response data was then transformed into a transactional dataset where each student represents a 'transaction,' and each incorrect answer represents an 'item' [6]. This binary transformation allows the algorithm to process "errors" as items to be associated. A snippet of this transformed data is presented in Table 2.

Table 2 - Transformation of Student Responses into Transactional Dataset

Name	Question_1	Question_2	Question_3	...	Question_59	Question_60
rpl1	0	0	1	...	0	1
rpl2	1	1	0	...	0	1
rpl3	0	0	0	...	1	1
rpl4	0	1	1	...	1	1
rpl5	0	0	1	...	1	1
...	...	...	...	...	...	...
rpl35	1	1	0	...	1	1

2. **Model Construction:** The transactional dataset was processed using the Apriori algorithm within the R programming environment, specifically leveraging the arules package [18]. The algorithm generated association rules based on two calibrated parameters: *Support* (set at 70%) and *Confidence* (set at 80%). Given the small sample size (n=35), a high support threshold (70%) was deliberately selected to ensure that the extracted rules represent difficulty patterns experienced by the vast majority of the class (approx. 25 students or more), rather than isolated incidents. Meanwhile, the 80% confidence threshold ensures high predictive reliability [8].

3. **Interpretation and Intervention:** The generated rules were filtered based on the *Lift* metric (*Lift* > *1*) to identify meaningful correlations rather than coincidental ones [6]. These insights were then used to design the specific remedial intervention using the Scaffolding model assisted by Block-Based Programming, which was subsequently tested for effectiveness using a post-test.

*Corresponding author: wahyudin_sanusi@upi.edu

3. RESULT AND DISCUSSION

This section presents the empirical findings of the study, structured into two phases: the diagnosis of learning difficulties using Association Rule Mining (ARM) and the evaluation of the subsequent scaffolding intervention.

3.1 Mining Cognitive Barriers: The "Domino Effect" in OOP

The core analysis began by processing the pre-test data of 35 students using the Apriori algorithm. To balance sensitivity and significance, the parameters were calibrated at a minimum support of 70% and a confidence of 80%. This initial process generated a massive set of 4,218 association rules.

However, to isolate meaningful learning patterns from trivial correlations, a rigorous filtering process was applied using the *Lift* metric. Only rules with a Lift > *1.1* were retained. This statistical filtering significantly reduced the dataset to 43 significant rules, which represent the core difficulty patterns found in the classroom.

Table 3 presents the summary of these 43 significant rules in their numerical format. Each number corresponds to a specific test item ID answered incorrectly by the students.

Table 3 - Example of Association Rules in Numerical Format

No	Rules	Support (A,B)	Support (A)	Confidence P(B|A)	Lift
1	8,25 ⇒ 14	80.00%	82.90%	96.60%	1.126437
2	43,56 ⇒ 38	80.00%	82.90%	96.60%	1.126437
3	30,43 ⇒ 38	80.00%	82.90%	96.60%	1.126437
4	26,43 ⇒ 38	80.00%	82.90%	96.60%	1.126437
…	…	…	…	…	…
38	7,26,31 ⇒ 38	82.90%	85.70%	96.70%	1.127778
39	19,59,60 ⇒ 45	80.00%	80.00%	100.00%	1.129032
40	19,24,59 ⇒ 45	80.00%	80.00%	100.00%	1.129032
…	…	…	…	…	…
43	17,24,59 ⇒ 45	82.90%	82.90%	100.00%	1.129032

As shown in Table 3, the resulting 43 rules exhibit high confidence values, validating the statistical strength of the relationships. However, raw numerical associations are insufficient for educational diagnosis. Therefore, a secondary filtering phase was conducted to translate these numerical patterns into actionable pedagogical insights.

Specifically, the 43 rules were filtered based on two main criteria: (1) No Item Redundancy, ensuring there is no overlap between the antecedent (cause) and the consequent (effect); and (2) Conceptual Soundness, requiring the rules to follow the logical hierarchy of OOP material. Pedagogically irrelevant rules, such as a difficulty in an advanced topic causing difficulty in a basic one (backward causality), were eliminated to maintain diagnostic logic.

This rigorous selection process resulted in 14 final rules that comprehensively illustrate the "Cognitive Domino Effect." These rules, which map specific prerequisite failures to subsequent concept breakdowns, are presented in Table 4.

Table 4 - Set of Association Rules for Learning Difficulty Patterns (Filtered)

No	Rules	Support (A,B)	Support (A)	Confidence P(B|A)	Lift
1	{Class & Object & Conclusion Drawing & Encapsulation & Argumentation Skill} ⇒ {Inheritance & Conclusion Drawing}	80.00%	82.90%	96.60%	1.165279
2	{Encapsulation & Sequential Thinking & Argumentation Skill} ⇒ {Inheritance & Conclusion Drawing}	85.70%	88.60%	96.80%	1.129032

*Corresponding author: wahyudin_sanusi@upi.edu

No	Rules	Support (A,B)	Support (A)	Confidence P(B|A)	Lift
3	{Class & Object & Conclusion Drawing & Encapsulation & Sequential Thinking} ⇒ {Inheritance & Conclusion Drawing}	85.70%	88.60%	96.80%	1.129032
4	{Encapsulation & Argumentation Skill & Class & Object & Sequential Thinking} ⇒ {Inheritance & Conclusion Drawing}	82.90%	85.70%	96.70%	1.127778
5	{Encapsulation & Sequential Thinking} ⇒ {Inheritance & Conclusion Drawing}	82.90%	85.70%	96.70%	1.127778
…	…	…	…	…	…
12	{Class & Object & Argumentation Skill & Encapsulation} ⇒ {Inheritance & Conclusion Drawing}	80.00%	82.90%	96.60%	1.126437
13	{Class & Object & Argumentation Skill & Conclusion Drawing & Encapsulation} ⇒ {Inheritance & Conclusion Drawing}	80.00%	82.90%	96.60%	1.126437
14	{Encapsulation & Sequential Thinking & Conclusion Drawing & Class & Object} ⇒ {Inheritance & Conclusion Drawing}	80.00%	82.90%	96.60%	1.126437

Table 4 reveals a consistent and striking pattern: the consequence for all significant rules is {Inheritance & Conclusion Drawing}. This indicates that the ultimate bottleneck for students is the ability to draw logical conclusions about Inheritance relationships.

Rule #1 (Lift 1.165) demonstrates that a combined weakness in *Class & Object* and *Encapsulation* (specifically in Argumentation and Conclusion skills) predicts failure in Inheritance with 96.6% confidence. Furthermore, Rule #5 highlights that even a specific deficiency in *Encapsulation & Sequential Thinking* is enough to trigger this failure (96.7% confidence). This confirms that Encapsulation is a "conceptual gateway"; students who cannot follow the sequential logic of data hiding (Encapsulation) inevitably fail to deduce the behavior of child classes (Inheritance)

To visualize these structural dependencies, a network graph is presented in Figure 3.

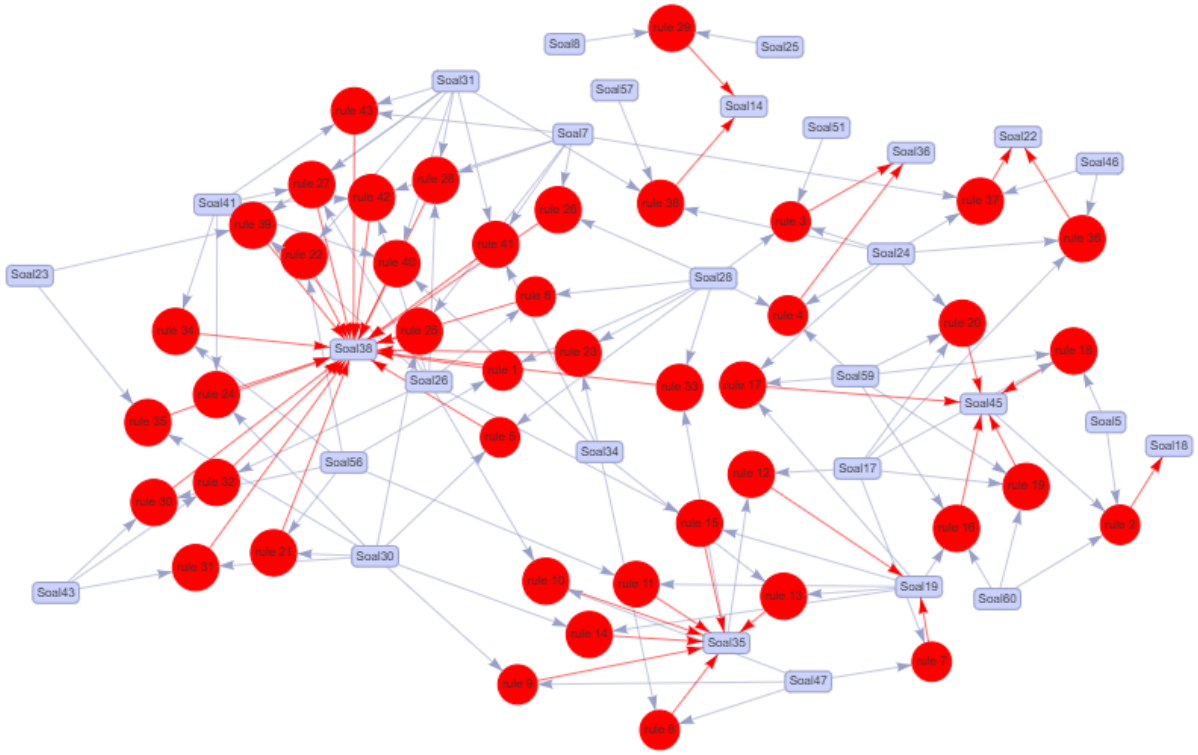


Figure 3 - Network visualization of student error patterns

In the graph in Figure 3, items related to basic concepts (e.g., questions about access modifiers in Encapsulation) serve as initial nodes with many outgoing arrows. Conversely, complex items like Item 38 (which tests Inheritance) act as difficulty "hubs," receiving numerous incoming arrows. This visually confirms that failure on Item 38 is not a random event but an accumulation of a series of prior conceptual failures, visual proof of the domino effect.

3.2. Efficacy of Scaffolding Intervention

Based on the diagnosis in Table 4, the remedial intervention was designed to break this chain of failure. The intervention utilized Block Programming to visualize the abstract concepts of *Encapsulation* and *Class & Object,* identified as the root causes in Rules #1 and #5.

The effectiveness of this targeted intervention was evaluated using a Paired Sample T-Test, as shown in Table 5.

Table 5 - Paired Sample T-Test Results

N	T	Df	Sig. (2-Tailed)
35	-12,14	34	<0,001

The analysis yields a significance value of $p < 0.001$, indicating a highly significant improvement. To further analyze the impact, N-Gain scores were calculated across student ability groups, as detailed in Table 6.

Table 6 - N-Gain Score Analysis by Student Group

Group	Average Pretest	Average Posttest	N-Gain	Category
Upper	69.58	84.17	0.34	Medium
Middle	82.73	85.45	-0.18	Low
Lower	74.58	92.92	0.71	High
Overall	75.43	87.57	0.30	Medium

Table 6 reveals that the intervention was most effective for the Lower Group (N-Gain 0.71), which aligns with the goal of scaffolding to support struggling learners. This significant improvement is visually summarized in Figure 4.

To visualize this significant leap in performance, a comparison chart is presented in Figure 4.

*Corresponding author: wahyudin_sanusi@upi.edu

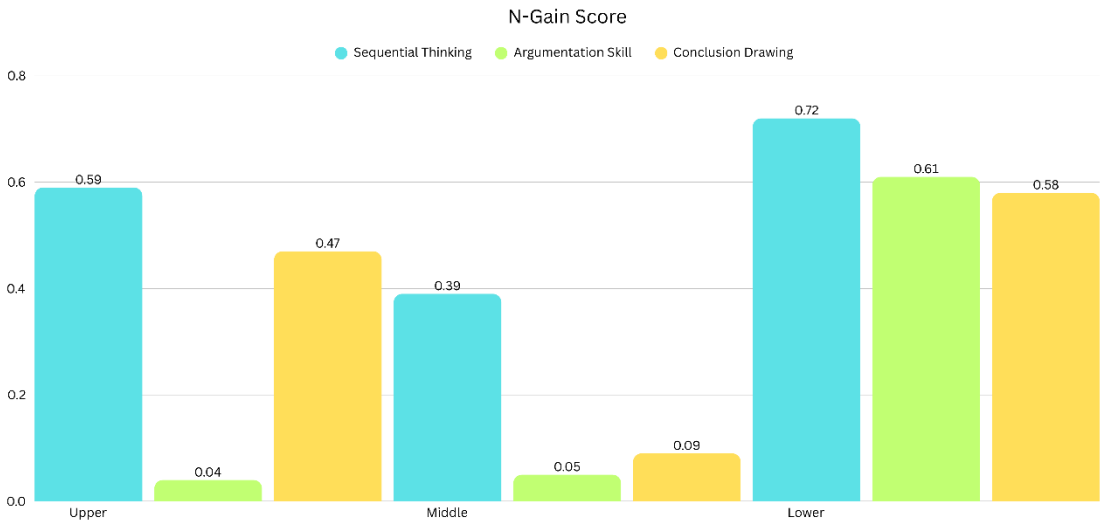


Figure 4 - Comparison of pre-test and post-test scores across student groups

The results in Figure 4 and Table 6 validate that diagnosing specific root causes (e.g., Encapsulation Sequential Thinking) allows for more effective remediation than generic re-teaching

The success of the Lower Group in Figure 4 validates Vygotsky's ZPD theory applied in this study. The diagnostic results from ARM allowed the teacher to provide specific scaffolding (e.g., focusing on Encapsulation blocks) exactly where the students were struggling. This contrasts with generic teaching methods that might treat all students equally. As noted by Zhou et al., adaptive scaffolding based on student needs is far more effective than "one-size-fits-all" instruction [12], [22].

4. CONCLUSION

This study successfully implemented an integrated framework combining Educational Data Mining and pedagogical scaffolding to address learning difficulties in Object-Oriented Programming (OOP). The research successfully achieved its two main objectives. First, the application of Association Rule Mining (ARM) on student pre-test data proved effective as a diagnostic tool. The analysis uncovered a "Cognitive Domino Effect," where failures in advanced topics are systematically rooted in specific prerequisite misconceptions. Specifically, the generated rules (e.g., Rule #1 and #9) identified with high confidence (*>96%*) that a lack of understanding in *Encapsulation* (particularly in sequential thinking and argumentation) acts as a primary barrier to mastering *Inheritance*.

Second, the remedial intervention utilizing a Scaffolding model assisted by Block-Based Programming was validated as highly effective. By targeting the specific "antecedents" identified by the ARM diagnosis (i.e., reinforcing Class & Object and Encapsulation concepts visually), the intervention significantly improved students' logical thinking skills ($p < 0.001$). The most notable impact was observed in the lower-achieving group, which experienced a "High" N-Gain increase of 0.71. This confirms that data-driven diagnosis, when paired with visual scaffolding, can effectively bridge the cognitive gap for novice programmers who struggle with abstract syntax.

However, this study has limitations that must be acknowledged. The research involved a relatively small sample size (n=35) from a single vocational school and employed a One-Group Pretest-Posttest design without a control group. Consequently, the generalizability of the findings may be limited. For future work, it is recommended to expand the scope using a larger sample size and a quasi-experimental design. Additionally, integrating this ARM-based diagnostic engine directly into an Intelligent Tutoring System (ITS) could automate the process of personalized feedback for large-scale vocational education settings.

Acknowledgement

The authors would like to express their gratitude to the principal, teachers, and students of SMK Negeri 1 Cimahi for their participation and support in this study.

*Corresponding author: wahyudin_sanusi@upi.edu

Appendix A: Sample of Logical Thinking Test Instrument

This appendix presents sample items from the 60-item instrument used in this study. The items vary from textual logic questions to visual block-to-code translation tasks, designed to measure specific cognitive aspects of OOP.

1. Topic: Encapsulation | Aspect: Argumentation Skill Question: A health application records patient data, including a medicalRecordNumber. To prevent unauthorized modification, this data must not be directly accessible or changeable from outside the class. Which strategy is best to maintain this data security? a. Make *medicalRecordNumber* a private attribute so it cannot be accessed directly. *(Correct Answer)* b. Make *medicalRecordNumber* public so it can be accessed anytime. c. Use a global variable so all classes can use it. d. Ensure *medicalRecordNumber* always contains a default value. e. Remove *medicalRecordNumber* from the class so it cannot be accessed.

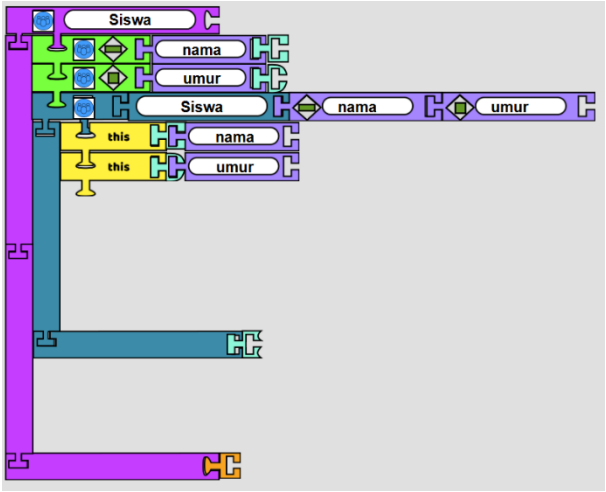
2. Topic: Class & Object | Aspect: Conclusion Drawing Question: In a racing simulation, a *Car* object has a *speed* attribute that increases if the *Accelerate()* method is called and decreases if the *Brake()* method is called. This change in speed happens dynamically according to the commands given. What conclusion can be drawn about the concept of objects in programming? a. Objects cannot change after they are created. b. Objects have a state that can change according to the methods called. *(Correct Answer)* c. All object attributes must be of type *const* so they do not change. d. Objects always have a fixed initial value. e. All objects in a class must have fixed values.

3. Topic: Class & Object | Aspect: Syntax-to-Visual Translation Question: Which of the following Block Programming structures correctly represents the Java code below?

```
public class Siswa
{
    public string nama;
    public int umur;

    public Siswa(string nama, int umur)
    {
        this.nama = nama;
        this.umur = umur;
    }
}
```

Correct Answer (Block Representation):



REFERENCES

[1] L. S. Vygotsky, *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press, 1978.

[2] OECD, *PISA 2022 Results (Volume I): The State of Learning and Equity in Education*. Paris: OECD Publishing, 2023.

[3] A. Aditomo and E. Klieme, "Forms of inquiry-based science instruction and their relations with science literacy: Evidence from the OECD PISA 2015 data," *Learning and Instruction*, vol. 68, p. 101330, 2020.

[4] C.-C. Hu and C. Ming-Hsien, "Enhancing learners' object-oriented programming skills through the integration of block model and UML class diagram," in *ICCE-Taiwan 2025 – 12th IEEE International Conference on Consumer Electronics*, 2025, pp. 495–496.

[5] S. Gomez-Jaramillo, O. Cardona-Zapata, M. Valbuena-Henao, and V. Poliche, "Guided learning with AI: A didactic strategy using sequential tutoring via ChatGPT for object-oriented programming," *International Journal of Learning, Teaching and Educational Research*, vol. 24, no. 7, pp. 717–736, 2025.

[6] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*, 4th ed. Waltham, MA: Morgan Kaufmann, 2022.

[7] R. Asif, A. Merceron, S. A. Ali, and N. G. Haider, "A systematic literature review on educational data mining strategies for identifying at-risk students," *Education and Information Technologies*, 2021.

[8] N. F. B. Casillano and K. W. Cantilang, "Employing educational data mining techniques to predict programming students at risk of dropping out," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 35, no. 2, pp. 1219–1226, 2024.

[9] G. Gao, S. Marwan, and T. W. Price, "Early performance prediction using interpretable patterns in programming process data," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE 2021)*, 2021, pp. 342–348.

[10] T. Talan, Y. Doğan, Y. Kalinkara, and V. Batdi, "To block or not to block?: A meta-analysis of the effectiveness of block-based programming," *Research in Science & Technological Education*, 2025.

[11] D. Jean, G. Stein, B. Broll, and A. Lédeczi, "Easing the block-to-text transition: A scaffolded approach to learning Python," in *Proceedings of the ACM Global Computing Education Conference (CompEd 2025)*, 2025, pp. 113–119.

[12] P. Zhou, Y. Li, Y. Zhang, Y. Yu, and Y. Tang, "The effects of scaffolding on learning outcomes in K-12 programming education: A meta-analysis," in *Proceedings of the 12th International Conference on Educational Innovation through Technology (EITT 2023)*, 2023, pp. 46–50.

[13] L. Zhang and J. Nouri, "A systematic review of research on computational thinking assessment in higher education," *Journal of Computing in Higher Education*, 2019.

[14] Y. Matsuzawa, Y. Ishikawa, and S. Sakai, "EnPoly: Workbench for understanding polymorphism in strongly typed object-oriented language," in *Proceedings of the 21st International Conference on Computers in Education (ICCE 2013)*, 2013, pp. 319–328.

[15] M. Tiantong and S. Teemuangsai, "The four scaffolding modules for collaborative problem-based learning through the computer network on Moodle LMS for the computer programming course," *International Education Studies*, vol. 6, no. 5, pp. 47–55, 2013.

[16] J. W. Creswell and V. L. P. Clark, *Designing and Conducting Mixed Methods Research*, 3rd ed. Thousand Oaks, CA: SAGE Publications, 2017.

[17] R. M. Branch, *Instructional Design: The ADDIE Approach*. New York, NY: Springer, 2009.

[18] M. Hahsler, S. Chelluboina, K. Hornik, and C. Buchta, "The arules R-Package Ecosystem: Analyzing Interesting Patterns from Large Transaction Datasets," *Journal of Machine Learning Research*, vol. 12, pp. 2021–2025, 2011.

[19] A. S. Nugroho and S. Wibowo, "Association Rule Mining for Analyzing Student Learning Patterns in an Online Programming Course," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 18, no. 4, pp. 133–148, 2023. *(Referensi pembanding sejenis)*

[20] Y. Chen, J. Liu, and L. Wang, "Identifying prerequisite relationships of programming concepts using association rule mining," *Journal of Educational Computing Research*, vol. 60, no. 2, pp. 425–448, 2022.

[21] M. Adraoui, A. Retbi, M. K. Idrissi, and S. Bennani, "Network visualization algorithms to evaluate students in online discussion forums," in *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, 2018, pp. 1–6.

[22] X. Gao, Y. Yang, Y. Du, and D. Sun, "Effect of Mind Mapping-Based Scaffolding on Elementary Students' Computational Thinking in Block-Based Programming," *J. Educ. Comput. Res.*, vol. 63, no. 1, pp. 236–271, 2025.

*Corresponding author: wahyudin_sanusi@upi.edu