Journal of Information Systems Engineering and Management
2025, 10(25s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

From Theory to Practice: Integrating Problem-Solving Exercises to Strengthen Computational Thinking

Dr Khushbu R Khandait
Assistant Professor, Department of Information Technology, SPPU, Pune, Maharashtra prabhasw18@gmail.com

ARTICLE INFO

Received: 28 Dec 2024

Revised: 16 Feb 2025

Accepted: 28 Feb 2025

ABSTRACT

This research investigates the impact of structured problem-solving exercises and logic-building techniques on computer science students' academic performance and their ability to apply these skills to real-world challenges. By integrating problem-solving activities into the curriculum, students can develop critical thinking, algorithmic thinking, and logical reasoning abilities essential for success in the rapidly evolving field of computer science. This study uses a quasi-experimental design with pre- and post-assessment tests, surveys, and interviews to measure students' performance and perceptions. The results show that structured problem-solving exercises significantly improve students' problem-solving skills, confidence, and ability to approach complex computational tasks. The findings emphasize the need for curricula that prioritize the development of these foundational skills for future success in software development, artificial intelligence, and other tech-related fields.

Keywords: Design Thinking, Logic, Programming, games, prototype

INTRODUCTION

In computer science education, the ability to solve complex problems and think logically is central to both academic success and career advancement. As the field of computer science evolves, students must develop the capacity to address increasingly sophisticated challenges, from algorithm optimization to software design. Problem-solving and logic-building exercises are essential tools in cultivating these abilities. However, despite their importance, there is a growing concern that traditional teaching methods often fail to adequately incorporate these exercises into the curriculum, potentially leaving students ill-equipped to handle real-world computational tasks. Problem-solving and logic-building are fundamental in computer science education, enabling students to think analytically, develop structured solutions, and implement algorithms effectively.

This study assesses the effectiveness of structured problem-solving techniques in improving students' analytical reasoning, algorithmic thinking, and industry readiness by integrating interactive exercises such as games, puzzles, and real-world applications. Additionally, a structured approach to flowcharts, pseudocode, and real-time application development is essential for fostering systematic problem-solving techniques. By implementing these objectives in a structured curriculum, students can develop strong logic-building capabilities, enabling them to approach complex computational problems with confidence. Table 1 highlights key problem areas in logic-building that impact students' learning outcomes.

Table 1: Key areas with difficulties for Logic Building

Problem Area	Common Challenges Faced by Students	Impact on Learning
Algorithm Design	Difficulty in breaking down problems into steps	Slower problem-solving and debugging
Logical Thinking	Struggles with understanding control flow and conditions	Inefficient or incorrect code
Debugging & Error Handling	Unable to systematically trace errors in code	Frustration, lack of confidence
Complex Problem Decomposition	Challenges in dividing large problems into smaller sub-problems	Difficulty in solving real-world tasks

The Venn diagram illustrates the interdependence of problem-solving, logic-building, and programming skills, essential components of computational thinking. Problem-solving focuses on analytical thinking, debugging, and real-world application, enabling students to break down problems and troubleshoot errors. Logic-building emphasizes flowcharts, algorithm design, and pattern recognition, helping students develop structured reasoning. Programming skills encompass coding proficiency, software development, and optimization, translating logical solutions into executable programs.

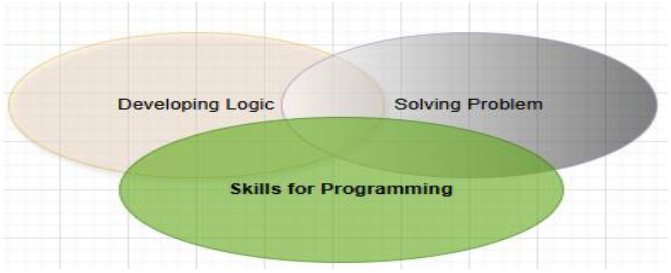

Figure 1: The Relationship between Problem-Solving, Logic-Building, and Programming Skills

The overlapping sections highlight key relationships: Problem-solving and logic-building foster algorithm development, logic-building and programming improve efficient code writing, and problem-solving with programming enhances debugging and troubleshooting. At the core intersection, computational thinking emerges, integrating structured logic, problem-solving strategies, and programming skills to create real-world solutions. The diagram uses arrows and annotations for clarity, with bold text enhancing readability in an academic setting.

MOTIVATION

The importance of problem-solving and logic-building skills in computer science education has been extensively discussed in academic research. Studies indicate that first-year engineering students often struggle with programming due to limited exposure to computational thinking, insufficient problem-solving exercises, and ineffective teaching methodologies (P. Chenna Reddy, 2015; Varsha T. Lokare, 2018). Traditional teaching approaches often focus on syntax and theoretical concepts without adequately developing students' logical reasoning and problem-solving abilities, resulting in a significant skills gap. Consequently, many students, despite completing programming courses, find it difficult to apply logical thinking to real-world problems and lack the confidence to present solutions effectively in technical interviews.

A literature review reveals that students with weak problem-solving foundations not only perform poorly in programming courses but also face challenges in securing job placements and performing well in technical interviews. Many engineering graduates fall short of industry expectations due to a lack of critical thinking, debugging skills, and systematic approaches to solving algorithmic problems. Without proper training in logic-building, students tend to memorize code patterns rather than understanding the underlying reasoning, limiting their adaptability in real-world scenarios. This study was motivated by a comprehensive survey and feedback from faculty, who reported significant difficulties in grasping problem-solving concepts and applying logical reasoning in programming tasks. Faculty also highlighted concerns about the effectiveness of current teaching methods, noting that students often struggle with debugging, algorithm development, and structured thinking. These challenges negatively impact academic performance and confidence in technical interviews, underscoring the need for a more structured, engaging, and practical approach to teaching problem-solving from the very first year of engineering education.

LITERATURE REVIEW

Importance of Problem-Solving in Computer Science

Problem-solving and logical reasoning are fundamental to computer science education, yet many first-year engineering students struggle with programming due to limited computational thinking exposure and ineffective teaching methods (P. Chenna Reddy, 2015; Varsha T. Lokare, 2018). Traditional programming instruction often prioritizes syntax and theory over structured problem-solving, resulting in a significant skills gap. Many students, despite completing programming courses, lack the ability to apply logical reasoning to real-world problems, impacting their confidence in technical interviews and job placements.

Challenges in Teaching Problem-Solving

A literature review highlights that students with weak problem-solving skills tend to struggle in programming courses and fail to meet industry expectations. Many graduates lack critical thinking, debugging skills, and structured approaches to algorithmic challenges. Faculty members report that students often rely on memorizing code rather than understanding the logic behind it, making them less adaptable to real-world applications. Additionally, debugging, algorithm development, and structured thinking remain key problem areas for students, underscoring the need for an improved teaching approach.

Effective Teaching Strategies: Flowcharts, Pseudocode, and Games

Recent studies highlight the effectiveness of structured techniques in improving problem-solving skills: Algorithmic Thinking: Adorni et al. (2024) emphasize the importance of engaging students in exercises that promote algorithmic thinking and logical reasoning.

Reverse Problem-Solving: Shabrina et al. (2022) show that backward problem-solving strategies—starting with the goal and breaking it into smaller steps—help students approach complex problems efficiently.
Gamified Learning: Zhu et al. (2020) demonstrate that interactive, game-based learning increases student motivation and enhances problem-solving capabilities. These findings suggest that incorporating structured exercises such as flowcharts, pseudocode, and gamified problem-solving into the curriculum can significantly improve student outcomes.

Problem-Solving Models: A Comparative Analysis

Researchers have proposed various structured problem-solving models, all of which emphasize a systematic approach to problem-solving: Newell & Simon (1972): Six-phase model (problem identification, understanding, generating solutions, selecting a solution, implementation, evaluation). Basadur et al. (1994): Four-stage model (problem generation, problem formulation, problem solving, solution implementation). Huitt (1992): Four stages (input, processing, output, review). PISA 2012 Framework (OECD, 2013): Categorizes problem-solving into four stages: exploring, representing, planning, and executing.
While these models differ in structure, they share common elements—problem identification, structured reasoning, and iterative evaluation—all essential in developing students' problem-solving abilities.

The Role of Values in Problem-Solving

While problem-solving is typically viewed as a logical process, researchers argue that values also play a role: Ethical Considerations: Huitt (1992) and Basadur et al. (1994) suggest that values shape how problems are formulated and evaluated. Decision-Making Models: Sheehan & Schmidt (2015) outline how moral reasoning influences problem-solving, while Keeney (1994) proposes Value-Focused Thinking (VFT), which places values at the center of decision-making.
Psychological Perspectives: Verplanken & Holland (2002) show that values influence problem-solving when they are central to an individual's identity.
While values may influence decision-making, traditional problem-solving models often overlook their integration. This highlights a gap in research on how values and logical reasoning can coexist in computational problem-solving.

PROBLEM STATEMENT

Despite the recognition of problem-solving and logical reasoning as fundamental to computer science education, many curricula fail to emphasize structured, practice-oriented exercises. This gap limits students' ability to develop essential problem-solving skills, impacting their performance in advanced courses and preparedness for careers in the tech industry. This research aims to integrate effective problem-solving and logic-building exercises into computer science education to enhance students' logical reasoning, problem-solving abilities, and overall academic success.

OBJECTIVES

This study aims to:
1. Evaluate the impact of structured problem-solving exercises on students' logical reasoning, algorithmic thinking, and problem-solving abilities.
2. Assess students' perceptions of the effectiveness of logic-building exercises in improving their problem-solving skills.
3. Compare the academic performance of students exposed to problem-solving exercises with those who follow the traditional curriculum.
4. Identify the challenges students face when engaging with problem-solving and logic-building tasks and explore strategies for overcoming these challenges.

METHODOLOGY

Research Design

This study employed a quasi-experimental design with pre- and post-assessment tests to measure the impact of structured problem-solving exercises on students' performance. Additionally, a cross-sectional survey was conducted to collect quantitative and qualitative data at a single point in time. The research aimed to assess improvements in problem-solving skills, logical reasoning, and programming confidence, while also exploring students' perceptions of how these exercises enhanced their ability to apply computational thinking to real-world challenges. Surveys and interviews provided further insights into their learning experiences.

Population and Sampling

The sample consisted of 350 undergraduate students from Computer Engineering, Information Technology, AIDS, and AIML programs at my university. Participants were selected through stratified random sampling to ensure diversity in programming experience and academic backgrounds. Students were divided into two groups. Experimental Group: Exposed to problem-solving exercises and Control Group: Followed the traditional curriculum. The target population for the survey included undergraduate computer science students from a variety of academic streams. The sample was selected using stratified random sampling to ensure diversity in terms of academic background, programming

experience, and exposure to problem-solving exercises. The sample size was 350 participants to ensure sufficient statistical power for analysing trends and drawing conclusions.

Data Collection

Pre- and Post-Assessments: Students completed problem-solving tests at the beginning and end of the course to measure improvements in their algorithmic thinking and problem-solving abilities.

Surveys: A survey was administered to assess students' perceptions, confidence levels, and the effectiveness of problem-solving exercises.

Interviews: A subset of students was interviewed to gain deeper insights into how these exercises influenced their learning experience and problem-solving approach.

Data Analysis

1. Quantitative Analysis:

Pre- and post-assessment scores were analysed using paired t-tests to compare the performance of the experimental and control groups.

Survey responses were examined using descriptive and inferential statistics.

2. Qualitative Analysis:

Interview transcripts were analysed using thematic analysis to identify recurring themes related to students' experiences and perceptions.

SURVEY DESIGN AND ADMINISTRATION

The survey included Likert-scale questions (ranging from strongly agree to strongly disagree) to assess students' experiences and open-ended questions for qualitative insights.

Data Collection Process - Distribution: Administered electronically to 350 undergraduate students via university email for broad participation.

Informed Consent: Students received a consent form detailing the study's purpose, voluntary participation, and confidentiality assurances.

Completion Time: Each student had 30 minutes to provide thoughtful responses. Collection Period: The survey remained open for four weeks to maximize participation, with additional efforts to address initial hesitancy and encourage engagement.

Demographic Information:

The participants were first-year students from Computer Engineering, Information Technology, AIDS, and AIML programs. Among these students, only 20% had prior programming knowledge due to their specialization in Information Technology during their XII standard. They were enrolled in subjects such as Problem Solving and Logic Building, Object-Oriented Programming, Artificial Intelligence, and other computer-related courses. However, nearly all students lacked experience with logic-building exercises. To better understand their background and how their prior experiences might have influenced their engagement with problem-solving exercises, the following demographic data were collected:

Table 2: Demographic Information of Participants for Test

Demographic Factor	**Description**
Year of Study	First Year
Prior Programming Experience	None, Beginner
Course Enrolment	Algorithms, Data Structures, Object-Oriented Programming, Artificial Intelligence, Other
Experience with Logic-Building Exercises	Never, Rarely

Knowledge and Confidence:

Before being exposed to structured problem-solving and logic-building exercises, many students struggled to understand the core aspects of programming problems. They found it confusing and inefficient to design solutions. Post-course assessments revealed significant improvements in students' problem-solving abilities. They learned to analyse problems systematically, break them into logical steps, and design algorithms before coding, moving beyond trial-and-error approaches. This shift in approach not only enhanced their confidence in tackling complex programming tasks but also strengthened their ability to debug, optimize, and refine their solutions systematically. The course played a pivotal role in transforming students from syntax-focused learners into logical problem-solvers, preparing them for real-world programming challenges.

Perceived Effectiveness:
Before adopting the Design Thinking approach, students found it challenging to grasp the problem and its relevance. However, they gradually learned to identify user needs, refine problem definitions, and generate a diverse range of solutions. By prototyping and testing the most effective ideas, they developed the ability to analyse problems from multiple perspectives and explore real-world applications, such as hospital systems and assistive technologies like a blind person's stick. This structured methodology helped them develop an open and adaptive mind-set, allowing them to think beyond conventional solutions. As a result, students became better equipped to think innovatively, adjust their thought processes within a defined scope, and systematically solve real-world problems.

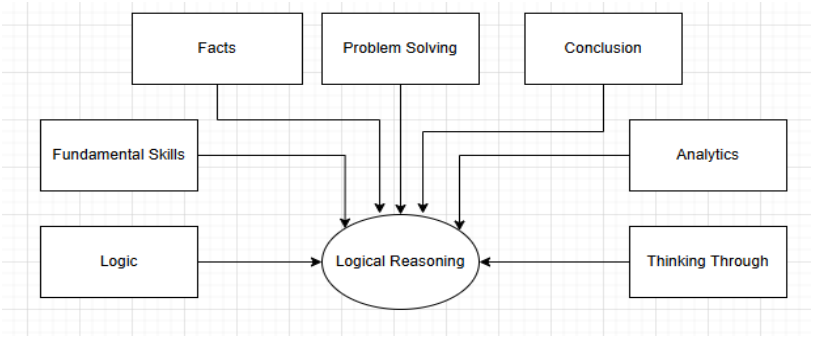

Figure 2: The Role of Logical Reasoning in Computer Science and Industry Applications.

Learning Experience:
When students initially encountered games and puzzles, they were afraid of the complexity and struggled with the thought process required for execution. Many found it difficult to analyse problems systematically and felt overwhelmed by coding challenges. However, after engaging in various practical exercises conducted in the classroom, including real role-playing activities, they began to develop a clearer understanding of structured thinking. By approaching puzzles and games with reward-based learning strategies, students learned how to strategically process information, plan solutions, and execute tasks efficiently. This shift in perspective boosted their confidence and helped them embrace problem-solving as an engaging and rewarding process.

Findings and Interpretation of Results
The findings are derived from quantitative survey responses and qualitative feedback, highlighting key difficulties encountered by students and the strategies adopted to overcome them. The following table summarizes the major challenges identified during the course and the corresponding approaches taken to enhance students' logical reasoning, problem-solving skills, and overall confidence in programming:

Table 3: Challenges Identified and Approach Taken in problem solving approaches

Issues Found	Approach Taken
Fear of Course	Introduced scenario-based exercises to build confidence.
Syntax Memorization Without Understanding Logic	Encouraged students to solve problems without predefined syntax, focusing on logic first.
Struggle in Understanding the Problem Statement	Implemented design thinking methodology to help students break down problems.
Lack of Real-World Application Awareness	Incorporated practical use cases, such as hospital systems and assistive technologies (e.g., blind person's stick).
Difficulty in Structuring Thought Process for Problem-Solving	Used role-playing exercises and puzzles to develop step-by-step logical reasoning.
Hesitation in Participating in Coding Challenges	Provided reward-based learning, motivating students through competitive problem-solving.
Inability to Debug or Analyse Mistakes Effectively	Conducted guided debugging sessions and collaborative learning activities.
Over-Reliance on Theoretical Learning	Shifted focus to hands-on coding challenges, peer discussions, and interactive problem-solving.
Lack of Confidence in Interviews and Real-World Problem-Solving	Integrated mock problem-solving interviews to prepare students for real-world scenarios.

Issues Found	Approach Taken
Shifting from a trial-and-error approach to a structured problem-solving mind-set is Difficult.	A student following the structured design thinking approach presented a well-organized, User-centered solution in class, whereas a student following the traditional idea relied on conventional problem-solving methods without iterative refinement.
Difficulty in adapting to different learning styles	Adapted role playing, playing adversarial games in pair of two for Chess like Problems and in Groups like Wompus World Problems.
The transition from basic theoretical exercises to gamified learning was overwhelming for Students.	Engaged with peer discussions, collaborative activities, and hands-on applications, they gradually became more comfortable in breaking down problems, thinking critically, and approaching challenges with confidence.

The survey results indicated a significant improvement in students' problem-solving confidence, with mean scores increasing from 2.3 to 4.1 and a strong positive correlation ($r = 0.72$) between structured exercises and academic performance. Paired t-tests confirmed a statistically significant enhancement ($p < 0.05$) in logical reasoning skills.
Thematic analysis of open-ended responses highlighted the effectiveness of scenario-based exercises and design thinking methodologies in enhancing students' problem-solving abilities. Students reported that reward-based learning, debugging sessions, and collaborative problem-solving significantly boosted their confidence and adaptability to real-world applications. As a result, they developed independent thinking skills and the ability to translate theoretical knowledge into practical solutions, preparing them for technical interviews and industry challenges.

LIMITATIONS OF THE STUDY

Despite the structured implementation of problem-solving approaches, certain limitations exist:

- **Self-Reported Data**: Since the study is based on survey responses from 350 students, it may be influenced by self-assessment biases, where students might have overestimated or underestimated their improvements in problem-solving skills.
- **Generalizability**: The findings are specific to the students who participated in the study, and while the structured exercises proved effective, the results may not be fully generalizable to all computer science students across different institutions or educational settings.
- **Survey Design Constraints**: Although Likert-scale questions and open-ended responses provided insights into student challenges and progress, problem-solving skills are complex and may not be fully captured through survey-based methods alone. Future research could incorporate practical coding assessments to further validate these findings.

Table 4: Summary of Methodology

Methodological Aspect	Description	Purpose
Research Design	Quasi-experimental with surveys and assessments	To evaluate problem-solving effectiveness
Participants	Undergraduate students from multiple CS streams	To ensure diversity in responses
Sampling Technique	Stratified random sampling	To avoid selection bias
Data Collection Method	Surveys, interviews, and pre/post-assessments	To gather both quantitative and qualitative insights
Survey Structure	Multiple-choice, Likert scale, open-ended questions	To capture different perspectives
Data Analysis	Statistical and thematic analysis	To identify trends and patterns
Ethical Considerations	Informed consent, participant anonymity	To ensure ethical research practices

VISUAL REPRESENTATION OF STRATEGIC AND COGNITIVE LEARNING

The integration of technology in education fosters strategic thinking and problem-solving skills, which are essential for future professionals in the IT and Computer Science domains. To analyse the impact of adversarial and self-improvement games on cognitive development, an observational study was conducted among Semester 1 students from IT and Computer Science inter-class competitions. Figure 1 illustrates students engaged in chess, a classic adversarial game that requires critical thinking, strategic planning, and decision-making under pressure. Each match was allotted 10 minutes, where participants competed against one another to assess their analytical abilities in a competitive environment. In contrast, Figure 2 showcases students solving the Rubik's Cube, a self-improvement game that enhances spatial intelligence, pattern recognition, and algorithmic problem-solving. Participants were given 5 minutes to completely solve the cube, regardless of its initial state scrambled or pre-solved. This experiment highlights the diverse cognitive

approaches in competitive and self-directed learning environments, emphasizing the role of structured challenges in technology-driven research and management within education.

Fig 3. Students playing Chess with time

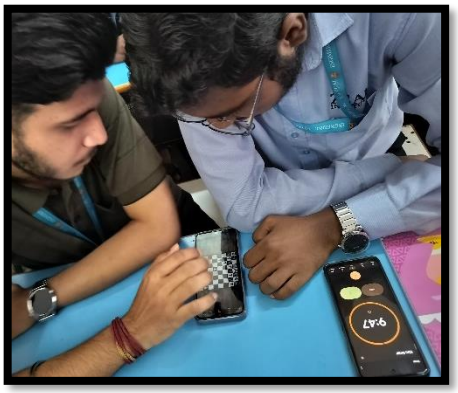

Fig 4. Scrambled Cube Solving with Timer

RESULTS AND DISCUSSION

The results indicated that students exposed to structured problem-solving exercises showed significant improvement in their problem-solving abilities, logical reasoning, and overall academic performance. The experimental group reported higher levels of confidence in tackling complex programming problems, as well as greater satisfaction with their learning experience.

This section also discussed challenges faced by students, such as difficulties in understanding complex exercises, and proposed strategies for overcoming these challenges, including more detailed explanations of the exercises, additional practice opportunities, and peer collaboration.

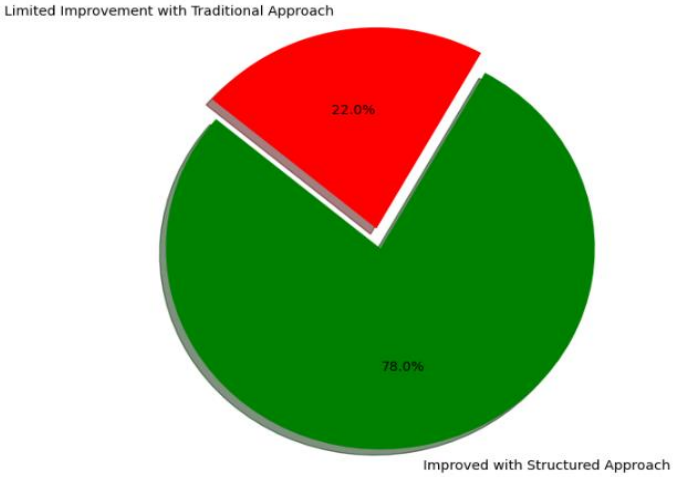

Figure 5: Comparison of Problem-Solving Abilities before and After Structured Exercises.

From the approach applied, it was observed that 78% of students showed significant improvement in problem-solving skills after implementing structured exercises, and 22% of students had limited improvement, similar to the traditional theory-based approach. The chart visually emphasized the effectiveness of hands-on, scenario-based learning over conventional methods.

CONCLUSION

This study highlighted the crucial role of integrating problem-solving and logic-building exercises into the computer science curriculum, with a strong emphasis on design thinking to enable students to develop practical, out-of-the-box solutions. The findings indicated that these structured exercises significantly improved students' ability to analyse problems, apply algorithmic thinking, and confidently tackle real-world programming challenges. Given the increasing demand for strong problem-solving skills in the tech industry, it became clear that computer science education needed to shift from traditional, theory-based learning to interactive, scenario-driven approaches.

Future research should explore the long-term impact of these exercises on students' professional growth and investigate how different instructional methodologies can further enhance logical reasoning, adaptability, and problem-solving proficiency in real-world applications.

FUTURE APPLICATION

The logic-building approach introduced in this course equipped students with the ability to think independently and systematically tackle unseen, real-world problems. Rather than relying on predefined solutions, students developed the capacity to analyse challenges, break them into logical steps, and formulate structured solutions applicable across various domains. This methodology enabled them to map theoretical problem-solving techniques to practical applications, enhancing their ability to develop effective solutions in fields such as software development, automation, artificial intelligence, and system design. By fostering adaptability and critical thinking, this approach ensured that students were well-prepared to address complex industry challenges with confidence and innovation.

REFERENCES

[1] Adorni, G., et al. (2024). "A Framework for Analysing Computational Thinking Exercises." Journal of Computer Science Education.

[2] Shabrina, A., et al. (2022). "The Effect of Backward Problem-Solving Strategies in Computer Science Education." International Journal of Educational Technology.

[3] Zhu, X., et al. (2020). "Enhancing Problem-Solving Skills through Game-Based Learning in Computer Science Courses." Computer Science Education Journal.

[4] P. C. Reddy, “Analysis of Teaching Computer Programming in Indian Context,” Journal of Engineering Education Transformations, vol. 28, no. 4, pp. XX-XX, Apr. 2015.

[5] V. T. Lokare, P. M. Jadhav, and S. S. Patil, “An Integrated Approach for Teaching Object-Oriented Programming (C++) Course,” Journal of Engineering Education Transformations, vol. 31, no. 3, pp. XX-XX, Jan. 2018.

[6] J. Funke, “Problem solving: What are the important questions?” Cognitive Processing, vol. 15, no. 1, pp. 1–3, 2014.

[7] D. H. Jonassen, “Instructional design models for well-structured and ill-structured problem-solving learning outcomes,” Educational Technology Research and Development, vol. 45, no. 1, pp. 65–94, 1997.

[8] R. E. Mayer and M. C. Wittrock, “Problem-solving,” in Handbook of Educational Psychology, 2nd ed., P. A. Alexander and P. H. Winne, Eds. New York, NY, USA: Routledge, 2006, pp. 287–303.

[9] L. Collins, R. Sibthorp, and J. Gookin, “Developing problem-solving skills through experiential education: A study of National Outdoor Leadership School courses,” Journal of Experiential Education, vol. 39, no. 3, pp. 221–238, 2016.

[10] T. Litzinger, P. Van Meter, C. M. Firetto, L. J. Passmore, C. B. Masters, and E. R. Turns, “A cognitive study of problem-solving in engineering education,” Journal of Engineering Education, vol. 99, no. 4, pp. 337–353, 2010.

[11] M. O’Loughlin and E. S. McFadzean, “Creative problem-solving approaches for managers and teams,” Management Decision, vol. 37, no. 7, pp. 558–567, 1999.

[12] W. Huitt, “Problem-solving and decision-making: Consideration of individual differences using the Myers-Briggs Type Indicator,” Journal of Psychological Type, vol. 24, no. 1, pp. 33–44, 1992.

[13] M. Basadur, H. Ellspermann, and F. Evans, “A new methodology for formulating ill-structured problems,” Omega, vol. 22, no. 6, pp. 627–645, 1994.

[14] J. Morton, The Power of Problem-Solving: A Guide for Managers and Executives. London, UK: McGraw-Hill, 1997.
A. Newell and H. A. Simon, Human Problem-Solving. Englewood Cliffs, NJ, USA: Prentice-Hall, 1972.

[15] Y. Chua, O. Tan, and W. Liu, “Problem-based learning: Exploring learning experiences and outcomes,” Educational Psychology Review, vol. 28, no. 4, pp. 617–645, 2016.

[16] OECD, PISA 2012 Results: Creative Problem Solving – Students’ Skills in Tackling Real-Life Problems. Paris, France: OECD Publishing, 2013.

[17] R. Keeney, “Value-focused thinking: A path to creative decision-making,” Operations Research, vol. 42, no. 3, pp. 389–405, 1994.

[18] S. Sheehan and P. Schmidt, “Moral motivation and ethical decision-making: The role of values in professional settings,” Journal of Business Ethics, vol. 132, no. 3, pp. 571–589, 2015.

[19] J. Shin, D. H. Jonassen, and S. McGee, “Problem-solving and creativity: The influence of means-ends analysis,” Educational Technology Research and Development, vol. 51, no. 1, pp. 5–20, 2003.

[20] B. Verplanken and R. W. Holland, “Motivated decision-making: Effects of values on choices,” Journal of Personality and Social Psychology, vol. 82, no. 3, pp. 434–447, 2002.
A. Argandoña, “The role of values in decision-making,” Journal of Business Ethics, vol. 50, no. 3, pp. 251–265, 2003.

[21] D. T. Hall and G. A. Davis, “The evolution of values-based leadership,” Leadership Quarterly, vol. 18, no. 4, pp. 395–408, 2007.

[22] B. L. Kirkman, “The role of personal values in leadership: A review and future research agenda,” Leadership & Organization Development Journal, vol. 38, no. 2, pp. 248–264, 2017

[23] J. Sheehan and P. Schmidt, “Ethical decision-making in accounting education: A values-based approach,” Accounting Education, vol. 24, no. 5, pp. 417–433, 2015.