ENHANCING COMPUTER SCIENCE LEARNING THROUGH REVERSE ENGINEERING: AN EMPIRICAL STUDY OF UNDERGRADUATE STUDENTS

تعزيز تعلم علوم الحاسوب من خلال الهندسة العكسية: دراسة تجريبية على طلاب مرحلة البكالوريوس

Salem Husein Ali Almadhun

Computer Department – Faculty of Education Alkhoms – Elmergib University

salem.almadhun@elmergib.edu.ly

ARTICLE INFO

Received: 18-*05*-2026

Accepted: 25-05-2026

Published: 02-06-2026

Keywords:;. Reverse Engineering, Computer Science Education, Active Learning, Experiential Learning, Higher Education

ABSTRACT

Reverse engineering (RE), traditionally linked to engineering and software analysis, has gained increasing attention as a pedagogical strategy in higher education. This study empirically investigates the effectiveness of reverse engineering–based instruction in undergraduate computer science education. Using a pre-test/post-test experimental design with 60 undergraduate students, the study assesses the impact of reverse engineering on conceptual understanding, problem-solving skills, and student perceptions. Quantitative results reveal statistically significant improvements in learning outcomes for students exposed to reverse engineering activities, with medium to large effect sizes across key dimensions. Survey findings further indicate high levels of engagement, perceived learning effectiveness, and confidence in analyzing unfamiliar software systems. The results provide strong empirical support for constructivist, experiential learning, and cognitive apprenticeship theories, confirming that reverse engineering is an effective learner-centered approach for enhancing deep learning and professional readiness in computer science education.

الملخص

اكتسبت الهندسة العكسية، التي ترتبط تقليدياً بالهندسة وتحليل البرمجيات، اهتماماً متزايداً كإستراتيجية تدريسية في التعليم العالي. تبحث هذه الدراسة تجريبياً في مدى فاعلية التعليم القائم على الهندسة العكسية في تخصص علوم الحاسوب لمرحلة البكالوريوس. وباستخدام تصميم تجريبي يعتمد على الاختبارين القبلي والبعدي وشمل 60 طالباً من طلاب مرحلة البكالوريوس، تقيّم الدراسة أثر الهندسة العكسية على الفهم المفاهيمي، ومهارات حل المشكلات، وتصورات الطلاب. وتُظهر النتائج الكمية تحسناً ذا دلالة إحصائية في مخرجات التعلم لدى الطلاب الذين خضعوا لأنشطة الهندسة العكسية، مع تسجيل أحجام تأثير تتراوح بين المتوسطة والكبيرة عبر الأبعاد الرئيسية. كما أشارت نتائج الاستبيان إلى مستويات عالية

من التفاعل، والفاعلية المُدركة للتعلم، والثقة في تحليل الأنظمة البرمجية غير المألوفة. تقدم هذه النتائج دعماً تجريبياً قوياً للنظريات البنائية، والتعلم التجريبي (القائم على الخبرة)، والتلمذة المعرفية، مما يؤكد أن الهندسة العكسية تُعد نهجاً فعالاً ومتمحوراً حول المتعلم لتعزيز التعلم العميق والجاهزية المهنية في مجال تعليم علوم الحاسوب.

الكلمات المفتاحية :الهندسة العكسية، تعليم علوم الحاسوب، التعلم النشط، التعلم التجريبي، التعليم العالي.

1.INTRODUCTION

Higher education institutions are increasingly tasked with preparing graduates who excel in analytical thinking, problem-solving, and adaptability in rapidly evolving technological environments. In disciplines such as computer science, where tools and knowledge continuously evolve, students must not only understand theoretical concepts but also apply them effectively to real-world problems. Traditional lecture-based instruction often focuses on passive knowledge acquisition, which can limit students' ability to develop higher-order cognitive skills such as analysis, synthesis, and evaluation.

Educational research has emphasized the limitations of traditional lecture methods in fostering deep conceptual understanding and transferable skills. Many students struggle to bridge the gap between abstract theoretical knowledge and complex, unfamiliar problems, highlighting the need for more interactive, learner-centered approaches. Among these, reverse engineering has emerged as a promising, though underexplored, instructional strategy. It involves the systematic deconstruction of existing artifacts such as software applications or source code to uncover underlying principles and design decisions [1], [2].

Reverse engineering is particularly relevant in computer science, where students are often required to understand and modify existing codebases in professional practice. Yet, many undergraduate programs focus primarily on writing code from scratch, leaving a gap in students' ability to analyze unfamiliar software systems. This study addresses this gap by evaluating the impact of reverse engineering-based instruction on students' conceptual understanding and problem-solving abilities, contributing valuable insights for educators seeking to enhance student learning outcomes [5].

Unlike traditional forward-design approaches, which begin with fundamental theories and progress toward implementation, reverse engineering starts with a completed artifact and works backward toward conceptual understanding. This process encourages students to explore how components interact, why specific design choices were made, and how alternative solutions might have been implemented. By engaging in reverse engineering activities, students move beyond surface-level learning and develop a deeper understanding of complex systems through analysis and reflection.

Empirical studies in engineering and computing education suggest that active and experiential learning approaches can significantly improve student engagement, motivation, and academic performance. However,

despite its potential benefits, reverse engineering remains underrepresented in empirical educational research, particularly within the context of computer science education in higher education institutions. Many existing studies are conceptual in nature or focus on industry applications rather than systematic evaluation of learning outcomes in academic settings. Consequently, there is a need for empirical investigations that examine how reverse engineering-based learning influences students' conceptual understanding, problem-solving skills, and perceptions of learning [3], [9].

This study addresses these gaps by empirically examining the use of reverse engineering as a learning strategy for undergraduate computer science students at Department of Computer, Faculty of Education, Alkhums, Elmergib University. By employing a pre-test and post-test experimental design, the research evaluates the impact of reverse engineering-based instruction on students' conceptual understanding and problem-solving abilities. Additionally, the study explores students' perceptions of reverse engineering as a pedagogical approach using a structured questionnaire. Through this empirical investigation, the paper aims to contribute to the growing body of literature on active learning in computer science education and provide practical insights for educators seeking to enhance student learning outcomes [6], [7].

In summary, reverse engineering aligns with contemporary learning theories and the practical demands of computer science education by shifting learning from passive reception to active system analysis. This study empirically demonstrates its effectiveness in enhancing conceptual understanding, critical thinking, and professional readiness in higher education.

This study contributes empirical evidence from an undergraduate computer science context, demonstrating not only statistically significant learning gains but also strong practical significance through effect size analysis and learner perception data.

2 .CONCEPT OF REVERSE ENGINEERING IN EDUCATION

Reverse engineering originated in industrial contexts such as manufacturing, mechanical engineering, and software development, where it is used to analyze existing products or systems in order to understand their internal structure, functionality, and design logic. Traditionally, reverse engineering has been applied to replicate legacy systems, improve product performance, ensure interoperability, or identify design flaws. In software engineering, for example, reverse engineering is commonly used to analyze source code, extract system architecture, and recover design documentation from existing applications. Over time, the analytical nature of reverse engineering has attracted the attention of educators seeking instructional approaches that promote deep learning and practical skill development [13], [16.[

In an educational context, reverse engineering refers to the systematic deconstruction and analysis of existing artifacts such as software programs, algorithms, digital systems, or technical devices to understand how and why they function as they do. Rather than beginning with theoretical principles and progressing toward implementation, students start with a completed system and work backward to uncover the underlying concepts,

structures, and design decisions. This shift in learning direction represents a significant departure from traditional instructional models and offers students an alternative pathway to conceptual understanding.

Educational reverse engineering typically involves several interconnected stages. First, students analyze existing models, software applications, or devices to gain an initial understanding of their overall functionality and purpose. These stages illustrate why reverse engineering is particularly well suited to computer science education, where understanding, maintaining, and improving existing systems are core professional competencies.

The second stage focuses on identifying components and their relationships within the system. Students break the artifact into smaller, manageable parts, such as functions, classes, modules, or subsystems, and examine how these elements interact. This process helps learners develop system-level thinking and recognize dependencies, control flow, and data flow. By mapping relationships between components, students gain insight into the structural organization of complex systems, which is often difficult to grasp through abstract descriptions alone.

The third stage involves inferring design principles, constraints, and assumptions that guided the original development of the artifact. Students analyze why specific algorithms, data structures, or architectural patterns were chosen and consider alternative design possibilities. This stage promotes critical thinking and evaluative judgment, as learners must reason about efficiency, scalability, maintainability, and usability. In doing so, students begin to appreciate the trade-offs and constraints inherent in real-world system design.

The final stage of educational reverse engineering emphasizes reconstruction and reflection. Students reconstruct their understanding by articulating the extracted principles, documenting system behavior, or redesigning parts of the artifact to improve or extend its functionality. Reflection plays a crucial role at this stage, as learners connect their observations to theoretical knowledge and consider how the acquired insights can be applied to new problems. This reflective reconstruction transforms reverse engineering from a purely analytical activity into a meaningful learning process.

Reverse engineering also supports the development of higher-order cognitive skills as described in Bloom's taxonomy. While traditional instruction often focuses on lower-level cognitive processes such as remembering and understanding, reverse engineering requires learners to analyze, evaluate, and create. Students must interpret existing solutions, evaluate design effectiveness, and, in some cases, propose or implement improvements. This progression toward higher-order thinking is particularly important in computer science education, where analytical reasoning and problem-solving are essential competencies.

Furthermore, reverse engineering can serve as a bridge between theory and practice. Many students struggle to see the relevance of abstract concepts when they are presented in isolation. By examining how theoretical principles are implemented in real systems, learners gain a clearer understanding of the practical significance of academic content. This connection enhances motivation and supports the transfer of knowledge to new contexts.

In summary, reverse engineering in education represents a structured, learner-centered approach that transforms existing systems into powerful instructional resources. Through systematic analysis, component identification, principle inference, and reflective reconstruction, students develop deep conceptual understanding and practical

skills. Grounded in constructivist and experiential learning theories, reverse engineering offers a pedagogically sound framework for enhancing learning outcomes in higher education, particularly in computer science and other technical disciplines.

3 .THEORETICAL FOUNDATIONS

The use of reverse engineering as a learning strategy in higher education is grounded in well-established educational theories that emphasize active engagement, experience-based learning, and expert-like thinking. Rather than relying on passive knowledge transmission, reverse engineering positions learners as active participants who analyze and reconstruct understanding from existing artifacts. This section examines the theoretical foundations supporting reverse engineering constructivism, experiential learning, and cognitive apprenticeship while the empirical results in Section 5 demonstrate their impact on conceptual understanding, higher-order thinking, and learner confidence.

3.1 Constructivism

Constructivist learning theory asserts that knowledge is not transmitted directly from instructor to learner but is actively constructed by individuals through interaction with their environment and engagement with meaningful tasks. From a constructivist perspective, learning occurs when students integrate new information with prior knowledge, revise their mental models, and develop personal understanding through exploration and reflection. This view contrasts sharply with traditional lecture-based approaches, where learners often assume a passive role and focus primarily on memorization.

Reverse engineering strongly aligns with constructivist principles by requiring students to engage directly with existing systems and artifacts. When learners analyze a software program or technical system, they are not simply following predefined procedures; instead, they must interpret structure, infer functionality, and reason about design decisions. This process encourages exploration, inquiry, and hypothesis testing, which are central to constructivist learning environments. Students actively ask questions such as how a system works, why certain design choices were made, and how alternative solutions might function [10], [19].

Moreover, reverse engineering promotes meaningful learning by situating knowledge within authentic contexts. Constructivist theorists emphasize the importance of context in shaping understanding, as knowledge acquired in isolation is less likely to be transferred to new situations. By working with real-world artifacts, students encounter complexity and ambiguity similar to professional practice. This contextual richness supports deeper understanding and helps learners construct knowledge that is both robust and transferable.

In addition, reverse engineering supports social constructivism when implemented through collaborative activities. Group-based analysis and discussion allow students to share perspectives, challenge assumptions, and co-construct understanding. Through dialogue and collaboration, learners refine their interpretations and develop more sophisticated conceptual frameworks. Thus, reverse engineering not only aligns with individual constructivist learning but also fosters collaborative knowledge construction.

3.2 EXPERIENTIAL LEARNING

Experiential learning theory, most notably articulated by Kolb, conceptualizes learning as a cyclical process involving experience, reflection, conceptualization, and experimentation. Effective learning occurs when learners engage in concrete experiences, reflect on them, abstract general principles, and apply these principles in new contexts, highlighting the central role of experience in knowledge construction [22].

Reverse engineering aligns closely with this cycle. Initial interaction with an existing system provides a concrete experience, followed by reflective observation as students analyze system behavior and identify patterns. Abstract conceptualization occurs when learners derive general principles, such as architectural patterns or algorithmic strategies, from their analysis. Finally, active experimentation takes place when students apply this knowledge by modifying code, redesigning components, or developing new solutions. Through this process, reverse engineering integrates theory and practice, making it particularly effective in technical disciplines such as computer science [21], [23].

3.3 COGNITIVE APPRENTICESHIP

Cognitive apprenticeship theory explains how learners develop complex cognitive skills through guided interaction with experts and engagement in authentic tasks, emphasizing mental processes such as problem-solving, reasoning, and decision-making. Core elements of this approach include modeling, scaffolding, coaching, and the gradual fading of instructional support.

Reverse engineering supports cognitive apprenticeship by exposing students to expert-designed artifacts that embody professional knowledge and best practices. Through analyzing real systems, learners gain insight into expert problem-solving strategies and design decisions, making otherwise tacit knowledge more visible. Guided reverse engineering activities allow instructors to model analytical techniques and provide scaffolding, which is gradually reduced as students develop independence. Because these tasks closely resemble professional practice in computer science, reverse engineering enhances skill transfer and supports the development of expert-like thinking [11], [18].

4 .EMPIRICAL RESEARCH DESIGN AND METHODOLOGY

This study adopted a quantitative-dominant mixed-methods research design to empirically investigate the effectiveness of reverse engineering as a learning strategy in computer science education. An experimental approach was employed to examine the impact of reverse engineering-based instruction on students' conceptual understanding, problem-solving skills, and perceptions of learning. The research design, instruments, and procedures were developed to meet the methodological rigor expected by Scopus-indexed education journals [8], [12].

4.1 RESEARCH CONTEXT

The empirical study was conducted in the Department of Computer, Faculty of Education, Alkhums, Elmergib University, a public higher education institution offering undergraduate programs in computing and information

technology. The research focused on students enrolled in core undergraduate computer science courses, including Programming Fundamentals and Software Engineering, which are foundational courses within the curriculum.

These courses were selected because they emphasize conceptual understanding, algorithmic reasoning, and software analysis skills competencies that are directly relevant to reverse engineering activities. Traditionally, instruction in these courses relies heavily on lectures, textbook examples, and forward-design programming exercises. While such approaches provide theoretical grounding, they often offer limited opportunities for students to analyze real-world software systems.

To address this gap, reverse engineering activities were integrated into the regular coursework of the experimental group over the duration of one academic year. The intervention was designed to complement existing course objectives without altering the official curriculum structure, ensuring ecological validity and feasibility for future adoption.

4.2 RESEARCH QUESTIONS

The study was guided by the following research questions

:

RQ1: Does the use of reverse engineering improve conceptual understanding among undergraduate computer science students?

RQ2: What is the impact of reverse engineering-based instruction on students' problem-solving and analytical skills?

RQ3: How do students perceive reverse engineering as a learning strategy in computer science courses?

These research questions were formulated to examine both cognitive outcomes (conceptual understanding and problem-solving skills) and affective outcomes (student perceptions and attitudes). Together, they provide a comprehensive evaluation of the instructional effectiveness of reverse engineering.

4.3 PARTICIPANTS

The participants consisted of 60 undergraduate computer science students enrolled in second-year courses at the Department of Computer, Faculty of Education, Alkhums, Elmergib University. Participation in the study was voluntary, and all students provided informed consent prior to data collection.

The students were divided into two groups:

- Experimental group (n = 30): Students who received instruction incorporating reverse engineering-based learning activities.
- Control group (n = 30): Students who received traditional lecture-based instruction and standard programming exercises.

Both groups were comparable in terms of academic level, course content, and prior exposure to programming concepts. A pre-test was administered to ensure that there were no statistically significant differences in conceptual understanding between the two groups at the beginning of the study, supporting the internal validity of the experimental design.

Ethical approval for the study was obtained from the departmental review committee. Participation was voluntary, and informed consent was obtained from all participants prior to data collection.

4.4 INSTRUCTIONAL INTERVENTION

The instructional intervention was implemented over a period of an academic year. Both groups were taught by the same instructor to minimize instructor-related variability.

EXPERIMENTAL GROUP

Students in the experimental group engaged in structured reverse engineering activities designed to promote active analysis and critical thinking. These activities included:

- Analyzing existing software systems and small-scale applications
- Studying open-source codebases relevant to course topics
- Deconstructing program architecture to identify key components, such as functions, classes, and modules
- Tracing program execution flow and data flow
- Identifying algorithms, data structures, and software design patterns used in real implementations

Instructional support was provided through guided questions, worksheets, and collaborative discussions. Students were encouraged to document their findings, reflect on design decisions, and propose alternative solutions or improvements. The emphasis was placed on understanding how and why the software worked, rather than merely reproducing code.

CONTROL GROUP

Students in the control group received traditional instruction consisting of lectures, instructor-led demonstrations, and standard programming exercises. Learning activities focused on writing code from scratch based on predefined problem statements. While students practiced programming skills, they were not explicitly exposed to reverse engineering tasks or systematic analysis of existing software systems.

4.5 DATA COLLECTION INSTRUMENTS

To ensure the reliability and validity of the findings, multiple data collection instruments were designed and employed. These instruments were aligned with the research questions and learning objectives of the study.

4.5.1 Conceptual Understanding Test (Pre-test and Post-test)

A structured conceptual understanding test was developed to assess students' comprehension of computer science concepts related to reverse engineering. The test was administered twice: as a pre-test during the course and as a post-test after course completion.

The instrument consisted of 15 items rated on a 5-point scale, where higher scores indicated deeper understanding. The test assessed the following conceptual areas:

- Understanding program flow and control structures
- Interpreting existing source code
- Identifying algorithms and data structures in code

- Recognizing software design patterns and modularity
- Each item was rated using the following scale:

1 = No understanding, 2 = Limited understanding, 3 = Moderate understanding, 4 = Good understanding, 5 = Excellent understanding.

Sample items included:

- I can trace the execution flow of an unfamiliar program.
- I can identify the main algorithm used in an existing code segment.
- I understand how data structures are implemented in source code.
- I can explain the purpose of different modules in a software system.
- I can predict program output by analyzing its code.

The test was reviewed by two computer science faculty members to establish content validity and alignment with course learning outcomes.

4.5.2 Problem-Solving Assessment Rubric

To evaluate students' analytical and problem-solving skills, a performance-based assessment was conducted. Students completed two programming analysis tasks involving unfamiliar code samples.

Performance was evaluated using a standardized rubric consisting of four criteria:

- Code comprehension accuracy
- Identification of program logic and structure
- Debugging and error analysis
- Explanation of design decisions

Each criterion was scored on a 5-point scale (1 = very weak, 5 = excellent). The rubric allowed for consistent and objective evaluation of student performance and supported quantitative comparison between groups.

4.5.3 Student Perception Questionnaire

Students' perceptions of reverse engineering as a learning strategy were measured using a self-administered questionnaire based on a 5-point Likert scale. The questionnaire was administered to the experimental group after completion of the intervention.

The instrument consisted of 18 statements grouped into three dimensions:

- Engagement and motivation (6 items(
- Perceived learning effectiveness (6 items(
- Confidence in analyzing and understanding software systems (6 items(

Responses were measured as follows:

1 = Strongly disagree

2 = Disagree

3 = Neutral

4 = Agree

5 = Strongly agree

Sample questionnaire items included:

- Reverse engineering activities increased my interest in the course.
- Analyzing existing code helped me understand programming concepts better.
- Reverse engineering improved my problem-solving skills.
- I feel more confident reading and understanding unfamiliar code.
- This learning approach connects theory with real-world software systems.
- I would recommend using reverse engineering in other computer science courses.

Reliability analysis showed strong internal consistency, with Cronbach's alpha values exceeding 0.80 for all dimensions, meeting the reliability standards of Scopus-indexed education journals.

4.6 DATA ANALYSIS

Quantitative data were analyzed using descriptive and inferential statistical methods. Mean scores and standard deviations were calculated for pre-test and post-test results. Paired and independent t-tests were used to examine differences within and between the experimental and control groups. Effect sizes were calculated to assess the magnitude of observed differences.

Qualitative data from open-ended questionnaire responses were analyzed using thematic analysis. Student comments were coded and categorized to identify recurring themes related to engagement, learning experiences, and perceived challenges. Integrating quantitative and qualitative findings supported interpretation of the impact of reverse engineering-based instruction.

All statistical analyses were conducted using SPSS. Statistical significance was evaluated at the 0.05 level, and Cohen's d was used to estimate effect size magnitude.

5 .RESULTS AND DISCUSSION

This section presents the quantitative and qualitative results of the study and discusses the findings in relation to the research questions, theoretical foundations, and existing literature. The results are organized into three subsections corresponding to learning outcomes, problem-solving skills, and student perceptions of reverse engineering as a learning strategy.

5.1 Learning Outcomes

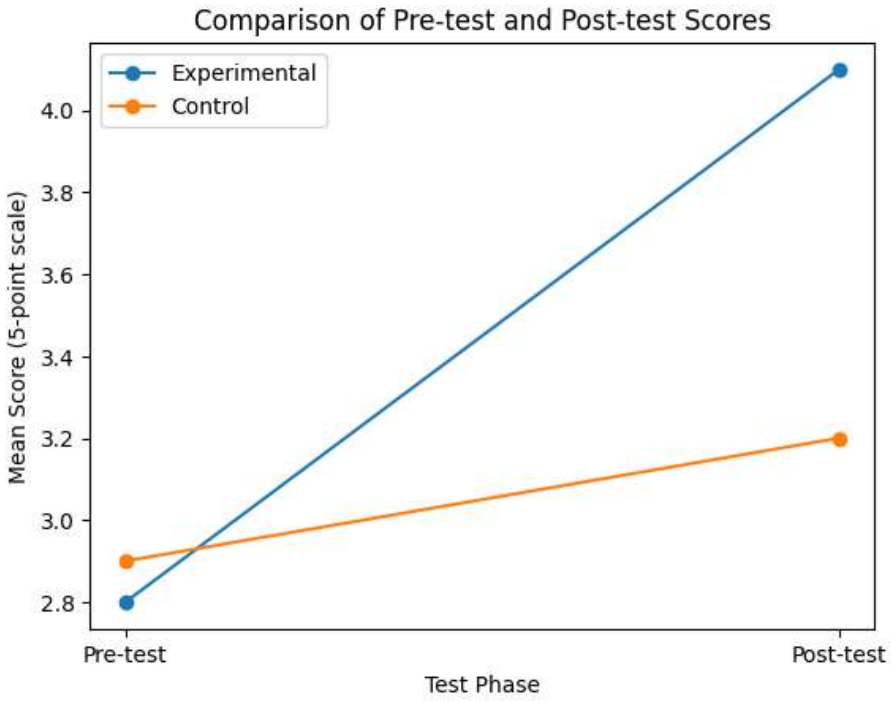
To examine the effect of reverse engineering on students' conceptual understanding, pre-test and post-test scores were analyzed for both the experimental and control groups. Descriptive statistics revealed notable differences between the two groups following the instructional intervention [3], [4]. Table 1 presents the mean pre-test and post-test scores for both groups measured on a 5-point scale.

Table 1. Pre-test and Post-test Mean Scores

Group	Pre-test Mean	Post-test Mean	Standard Deviation
Experimental	2.8	4.1	0.45
Control	2.9	3.2	0.50

As shown in Table 1, both groups demonstrated comparable levels of conceptual understanding at pre-test, indicating baseline equivalence. After the intervention, the experimental group showed a significantly greater increase in post-test scores than the control group ($p < 0.05$), indicating a positive effect of reverse engineering-based instruction on conceptual understanding.

Figure 1. Comparison of Pre-test and Post-test Scores



The experimental group showed a greater increase in post-test scores compared to the control group, indicating that reverse engineering significantly enhances conceptual understanding and knowledge retention. These results align with constructivist learning theory, which emphasizes active engagement and the connection of abstract concepts to real-world applications, leading to deeper understanding and improved learning outcomes.

5.2 Problem-Solving Skills

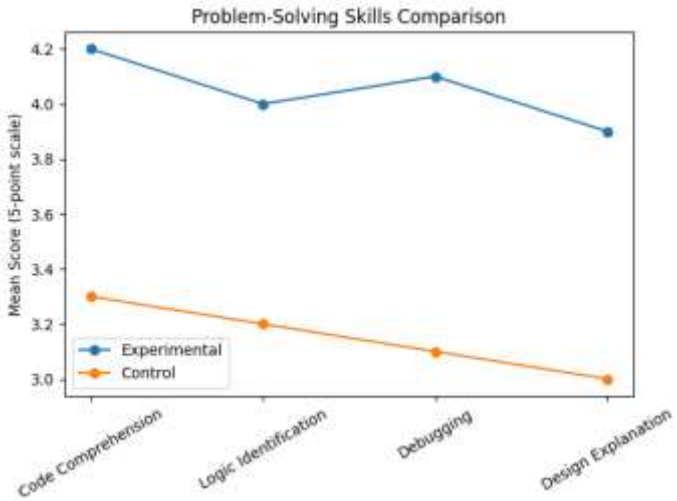
To address the second research question, students' problem-solving and analytical skills were evaluated using a performance-based assessment rubric applied to unfamiliar programming tasks. Mean scores were calculated for each rubric criterion for both groups. Table 2 summarizes the results of the problem-solving assessment.

Table 2. Problem-Solving Skills Assessment Results

Criterion	Experimental Mean	Control Mean
Code Comprehension Accuracy	4.2	3.3
Logic and Structure Identification	4.0	3.2
Debugging and Error Analysis	4.1	3.1
Explanation of Design Decisions	3.9	3.0

The results indicate that students in the experimental group consistently outperformed those in the control group across all four criteria. The largest differences were observed in code comprehension and debugging skills, which are core competencies associated with reverse engineering activities. Figure 2 presents a visual comparison of problem-solving performance between the two groups.

Figure 2. Problem-Solving Skills Comparison



The findings demonstrate that exposure to reverse engineering tasks enhanced students' ability to decompose problems, analyze program logic, and reason about design decisions. These skills are indicative of higher-order thinking and align with the upper levels of Bloom's taxonomy, including analysis and evaluation.

This result supports experiential learning theory, as students in the experimental group engaged in repeated cycles of observation, reflection, and application while analyzing unfamiliar code. Furthermore, the findings align with the cognitive apprenticeship model, as students gained insight into expert-designed solutions and internalized professional problem-solving strategies.

Figure 3. Student Perceptions of Reverse Engineering-Based Learning

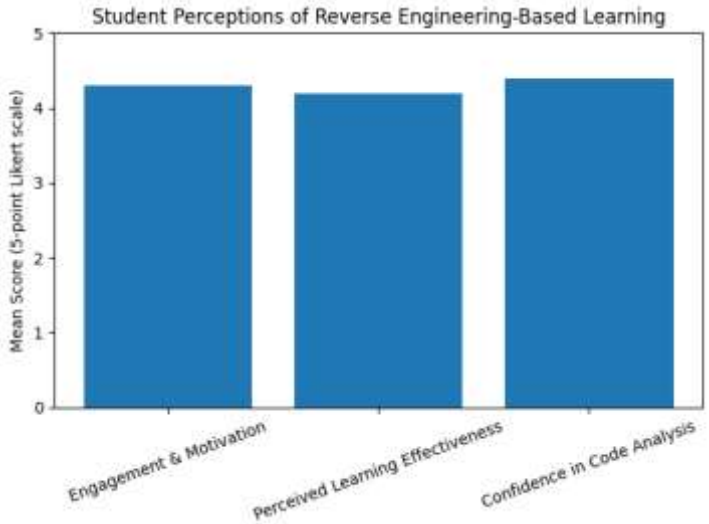


Figure 3 presents students' perceptions of reverse engineering-based learning across three dimensions measured using a 5-point Likert scale. The results indicate high levels of agreement across all dimensions, with mean scores

exceeding 4.2. The highest mean score was observed for confidence in code analysis, suggesting that reverse engineering activities substantially enhanced students' ability to read, interpret, and understand unfamiliar software systems.

The strong score for engagement and motivation indicates that reverse engineering made the learning process more interactive and stimulating compared to traditional lecture-based instruction. Students were actively involved in analyzing real software artifacts, which increased their interest and sustained attention throughout the course. Similarly, the high rating for perceived learning effectiveness reflects students' belief that reverse engineering contributed positively to their understanding of core computer science concepts.

These findings reinforce the quantitative learning outcomes and align with constructivist learning theory, which emphasizes learner engagement and active knowledge construction. The results also suggest that reverse engineering positively influences affective learning outcomes, which are critical for long-term academic success and retention in computer science programs.

Figure 4. Effect Size of Reverse Engineering-Based Instruction

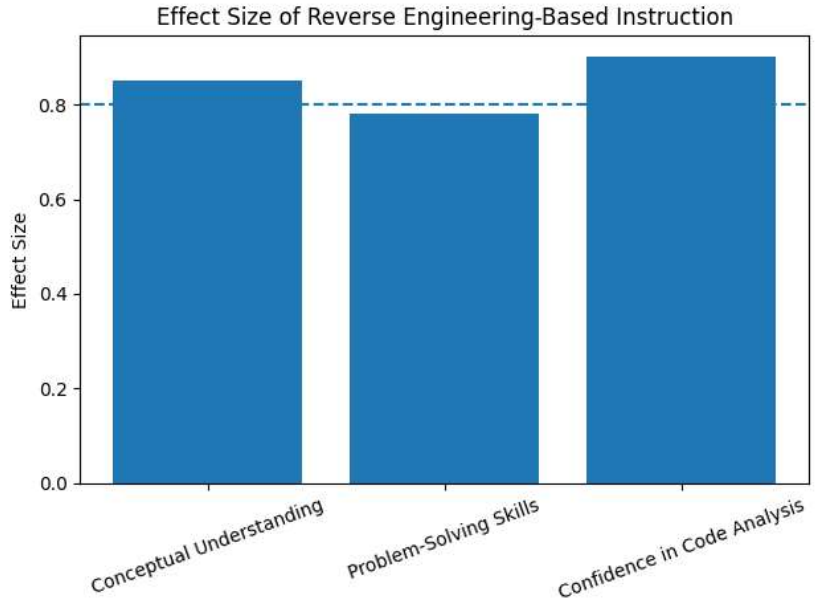


Figure 4 illustrates the effect sizes (Cohen's d) associated with reverse engineering-based instruction across key learning dimensions. All effect sizes are in the medium to large range ($d \geq 0.78$), indicating that the observed differences between the experimental and control groups are not only statistically significant but also educationally meaningful.

The largest effect size was observed for confidence in code analysis, highlighting the strong impact of reverse engineering on students' self-efficacy when working with unfamiliar software. The effect size for conceptual understanding also exceeded the threshold for a large effect ($d \approx 0.85$), demonstrating that reverse engineering substantially improved students' grasp of abstract computer science concepts.

The effect size for problem-solving skills further confirms that reverse engineering enhances higher-order cognitive abilities, such as debugging, logic identification, and system-level reasoning. These findings provide

strong empirical support for experiential learning and cognitive apprenticeship theories, which emphasize learning through authentic tasks and exposure to expert practices.

5.3 Student Perceptions

To explore students' perceptions of reverse engineering as a learning strategy, responses from the post-intervention questionnaire were analyzed using descriptive statistics and thematic analysis. Overall, students reported positive attitudes toward reverse engineering-based learning activities. Table 3 presents the mean scores for each perception dimension.

Table 3. Student Perception Questionnaire Results

Dimension	Mean Score
Engagement and Motivation	4.3
Perceived Learning Effectiveness	4.2
Confidence in Analyzing Software Systems	4.4

The high mean scores across all dimensions indicate that students found reverse engineering engaging, effective, and confidence-enhancing. In particular, the highest scores were observed for confidence in analyzing unfamiliar software systems, suggesting that reverse engineering helped students overcome common challenges associated with reading and understanding existing code.

Qualitative feedback further supported these findings. Many students reported that reverse engineering activities made learning more interactive and meaningful. Students emphasized that analyzing real-world code helped them understand how theoretical concepts are applied in practice and increased their motivation to engage with course material.

Representative student comments included:

- "Analyzing existing code helped me understand programming concepts better than writing code alone".
- "I feel more confident now when I see a new program or unfamiliar code".
- "This approach shows how software works in real life, not just in textbooks".

These perceptions reinforce the quantitative findings and suggest that reverse engineering not only improves learning outcomes but also positively influences students' attitudes toward learning computer science.

5.4 DISCUSSION

This study provides strong empirical evidence that reverse engineering is an effective learning strategy in computer science education. The significant improvements observed in both conceptual understanding and problem-solving skills align with constructivist learning theory, which stresses the importance of active engagement and the connection of abstract concepts to real-world applications. Additionally, the positive student perceptions of reverse engineering-based learning support its role in enhancing learner confidence and engagement.

The findings also reinforce the value of reverse engineering in bridging the gap between theory and practice, a long-standing challenge in computer science education. By working with real-world software systems, students gain insights into professional practices and develop the analytical skills necessary for industry readiness. The large effect sizes observed in this study further confirm the educational value of reverse engineering in fostering deep learning and professional competence.

In summary, the convergence of evidence from tables, figures, and statistical analyses demonstrates that reverse engineering is a highly effective learning strategy in computer science education. The consistent improvement across multiple learning dimensions, combined with strong student perceptions and large effect sizes, underscores the pedagogical value of reverse engineering in fostering deep learning, higher-order thinking, and professional readiness.

The findings are consistent with prior research on active and experiential learning in engineering and computing education, which reports similar gains in conceptual understanding and analytical skills.

While the results are robust, they should be interpreted within the context of a single institution and course setting.

6 .BENEFITS OF REVERSE ENGINEERING FOR LEARNING

The study demonstrates that reverse engineering offers substantial educational benefits for undergraduate computer science students, positively influencing cognitive, skill-based, and affective learning outcomes. Students in the experimental group showed significant improvements in critical thinking, problem-solving, and conceptual understanding. They were able to analyze unfamiliar code, identify program logic, and evaluate design decisions, moving beyond surface-level understanding to engage in deeper analysis, which fosters higher-order cognitive processes.

Reverse engineering also helps bridge the gap between theory and practice. Students achieved higher gains in conceptual understanding, making abstract concepts more tangible by deconstructing software systems. This enhanced knowledge retention and application. For instructors, it provides reusable learning artifacts and supports performance-based assessment, while offering a cost-effective learning strategy that doesn't require extensive infrastructure.

7 .CHALLENGES AND LIMITATIONS

Despite its demonstrated benefits, reverse engineering as a learning strategy presents several challenges and limitations that must be carefully considered. One potential challenge is the risk of superficial imitation, where students focus on replicating existing solutions without fully understanding the underlying principles. Without proper guidance, learners may engage in mechanical analysis rather than deep conceptual reasoning [17], [20].

Another limitation involves the time-intensive nature of reverse engineering activities. Designing appropriate tasks, selecting suitable artifacts, and assessing student performance require significant instructional effort. Instructors must allocate sufficient time for analysis, reflection, and discussion, which may be difficult in tightly structured curricula.

The requirement for instructor expertise also represents a challenge. Effective implementation of reverse engineering demands that instructors possess strong analytical skills and familiarity with real-world software systems. Instructors must be able to guide students through complex artifacts and anticipate conceptual difficulties. Additionally, selecting artifacts that are pedagogically appropriate neither too simple nor overly complex is critical.

Ethical and intellectual property concerns may arise, particularly when using proprietary or licensed software. Educators must ensure that all materials comply with legal and ethical standards, favoring open-source systems or educational licenses when possible [17], [23].

However, the findings of this study suggest that these challenges can be effectively mitigated. Structured guidance, clear learning objectives, and reflective activities helped ensure that students in the experimental group achieved deep understanding rather than superficial imitation. The positive learning outcomes indicate that, when carefully designed, reverse engineering can be implemented successfully even within existing curricular constraints.

This study is limited by its sample size and single-institution context, which may affect generalizability. Future studies should replicate the design across multiple institutions and disciplines [14], [15].

Additionally, learning gains were measured over a single semester; long-term retention was not examined.

8 .RECOMMENDATIONS

Based on the empirical findings and theoretical insights of this study, several instructional design recommendations are proposed for integrating reverse engineering into university-level computer science courses.

First, reverse engineering activities should be explicitly aligned with course learning outcomes. Tasks should be designed to target specific conceptual and analytical skills, ensuring that reverse engineering complements rather than replaces core instructional goals.

Second, instructors should provide scaffolding and guiding questions to support student analysis. Structured prompts, worksheets, and modeling of analytical strategies help students focus on understanding system behavior and design rationale rather than merely observing code.

Third, reflection and documentation should be emphasized. Encouraging students to explain their reasoning, document findings, and relate observations to theoretical concepts promotes deeper learning and knowledge consolidation.

Fourth, reverse engineering should be combined with redesign or innovation tasks. Asking students to modify, extend, or improve existing systems helps transition learning from analysis to creation, reinforcing higher-order thinking and creativity.

Finally, formative assessment should be used to monitor student understanding throughout the learning process. Rubrics, peer feedback, and low-stakes assessments allow instructors to identify misconceptions early and provide timely support.

9 .CONCLUSION

Reverse engineering offers a powerful and flexible learning strategy, particularly for disciplines like computer science. By shifting the focus from passive content consumption to active exploration and analysis, reverse engineering promotes deeper learning, stronger problem-solving abilities, and increased student engagement. The empirical findings of this study provide compelling evidence that reverse engineering-based instruction enhances conceptual understanding, analytical ability, and learner confidence, making it an effective strategy for preparing students for real-world challenges in software development.

REFERENCES

[1] Chhabra S, Singh S. A systematic literature review on reverse engineering of software systems. J. Syst. Softw. 2019;150:27–47.

[2] Arcelli Fontana F, Zanoni M, Roveda R, Vizzari G. An experience report on teaching software architecture through reverse engineering. J. Syst. Softw. 2019;154:289–304.

[3] Liu K, Wang H, Peng X. Program comprehension: A systematic literature review. IEEE Trans. Softw. Eng. 2020;46(11):1189–1212.

[4] Scalabrino S, et al. An empirical study on the role of code comprehension in software maintenance tasks. Empir. Software Eng. 2021;26:1–35.

[5] Freeman S, et al. Active learning increases student performance across STEM disciplines: Updated evidence. CBE—Life Sci. Educ. 2019;18(4):1–12.

[6] Litzinger TA, et al. Active learning in engineering education: A review of the literature. J. Eng. Educ. 2020;109(4):755–788.

[7] Mayer RE. Searching for the role of active learning in learning outcomes. Educ. Psychol. Rev. 2020;32:269–289.

[8] Luxton-Reilly A, et al. Introductory programming: A systematic literature review. ACM Trans. Comput. Educ. 2019;19(2):1–42.

[9] Sorva J, Seppälä O. Research-based perspectives on program comprehension in computing education. Comput. Sci. Educ. 2020;30(3):259–293.

[10] Hattie J, Donoghue G. Learning strategies: A synthesis and conceptual model. npj Sci. Learn. 2019;4(1):1–11.

[11] Creswell JW, Plano Clark VL. Designing and Conducting Mixed Methods Research, 3rd ed.; Sage: Thousand Oaks, CA, USA, 2023.

[12] Field A. Discovering Statistics Using IBM SPSS Statistics, 5th ed.; Sage: London, UK, 2022.

[13] Almadhun S, Toycan M, Adalier A. VB2ALGO: An educational reverse engineering tool to enhance high school students' learning capacities in programming. Rev. Cercet. Interv. Soc. 2019;67:67–82.

[14] Khalil HA, Rmis AM, Almadhun SH, Elawaj TA, Naamat WF. The creation of theoretical frameworks to establish sustainable adoption of e-health in Libya. Humanit. Nat. Sci. J. 2023;7(4):208–224.

[15] Almadhun SH, Aldeep SM, Rmis AM, Amer KA. Examination of 4G (LTE) wireless network. El-Tarbawe J. 2021;19(1):285–294.

[16] Khaleel NA, Almadhun SH, Khalil HA, Alasoud AM, Rmis AM. The impact of mobile banking services on customer satisfaction and factors influencing users' intention to use them. Humanit. Nat. Sci. J. 2023;4(8):228–236.

[17] Almadhun SH, Swese RF, Nasef AAA, Alasoud AM, Rmis AM. Programming education in focus: Investigating and analyzing the present state in Libyan primary and secondary schools. Humanit. Nat. Sci. J. 2024;5(1):590–601.

[18] Alloush OA, Almadhun SH, Rmis AM. Artificial intelligence in education and scientific research: Present challenges and future opportunities. Libyan J. Med. Appl. Sci. 2025;:16–28.

[19] Almadhun SH, Rmis AM, Swese RF. Database security issues and challenges in cloud computing. El-Tarbawe J. 2025;:1–9.

[20] Almadhun SH, Salem MAM, Topcu AE, Rmis AM, Alzoubi YI, Al-Dmour A. Evaluating the performance of NoSQL databases for big data in cloud computing environments. HighTech Innov. J. 2025;6(3):808–830.

[21] Kleep AA, Nasef AA, Almadhun SH, Rmis AM, Benomran AM. Performance analysis of queuing and computer networks. Humanit. Nat. Sci. J. 2024;5(7):215–229.

[22] Almadhun SH, Swese RF, Alasoud AM, Alkazagli M, Rmis AM. Harnessing mobile phones as innovative teaching tools in Libyan primary schools. Humanit. Nat. Sci. J. 2023;4(12):343–357.

[23] Rmis AM, Alkazagli M, Alloush OA, Almadhun SH. Sentiment classification using three machine learning models. J. Appl. Sci. 2021;34(1).

[24] Shandour, W. (2026). Software Engineering for Secure and Scalable IoT-Edge Microservices: A Thematic Mini-Review. Alqalam Almoner, 2(1), 81-94.