

Teaching Reform of the Data Structures and Algorithms Course Integrating the OBE Concept with AI Technology

Huihong Xie, Yinxing Chen *, Jiaying Wang, Yingshan Lin

Guangdong University of Science and Technology, Dongguan, Guangdong, 523000, China

* Corresponding author: Yinxing Chen

Abstract: Data Structures and Algorithms have for years run into the same wall: teaching that stays rigid, material too abstract for many students to grasp, a gap between what is taught and what is practiced, grades that lean almost entirely on a final exam, and little room for instruction shaped to the individual. To meet the talent demands of emerging engineering education, this study puts the OBE (Outcome-Based Education) idea together with artificial intelligence and rebuilds the course around them. Working backward from outcomes, the redesign sets course aims in three strands, knowledge, ability, and the broader qualities students should leave with. AI-driven visualization, learning analytics, and tutoring help build a learning space that straddles the online and the face-to-face, while a practice system that steps up in difficulty and an assessment that tracks students the whole way through keep the loop closed. The result moves the course away from handing over knowledge and toward the engineering competence graduates actually need, and it offers a model that other core engineering courses can adapt.

Keywords: OBE; Artificial Intelligence; Data Structures and Algorithms; Blended Teaching.

1. Introduction

Data Structures and Algorithms sits at the core of the curriculum for computer science, artificial intelligence, electronic information, and a number of adjacent engineering majors. It also supplies the groundwork for later courses, among them programming languages, operating systems, and big data analytics. What makes the course hard is not any single feature but the way it piles them together: abstract theory, tight logical reasoning, and programming work that demands real fluency. Small wonder it is routinely counted among the toughest parts of an engineering degree.

The conventional format still leans heavily on the lecture. An instructor works through the material at the board, exercises come afterward, and the textbook sets the pace from start to finish. The result is predictable. Sessions turn monotonous, interaction stays thin, and most of the harder structures and algorithms are shown only as text or static pictures. Students never actually watch an algorithm run. Faced with that, many give up on the underlying logic and simply memorize code templates, which leaves them poorly equipped to transfer anything to a real engineering problem.

We also find that the old course prizes theory at the expense of practice. The exercises on offer are few and narrow, and they rarely touch the kind of work that software development or data processing actually involves. Grading, meanwhile, rests almost entirely on a single written final. That exam rewards recall of the textbook and says little about how a student thinks, codes, or improvises. In short, the score is a weak proxy for what was actually learned.

Things have shifted. Emerging engineering education has pushed industry to ask for graduates who can work across disciplines and apply what they know, so hands-on skill and a habit of innovation now matter as much as raw knowledge [1]. OBE speaks directly to this. It puts student outcomes first and builds teaching backward from them, with student-centered instruction, backward design, and assessment that comes in more than one shape. AI adds a different kind of help. Through visualization, on-demand question answering,

learning analytics, and resource recommendations tuned to the individual, it targets exactly the points where this course tends to break down. Put the two together and the teaching system can be rebuilt around practice rather than transmission, moving the course toward the genuine engineering competence that graduates actually need.

2. Theoretical Basis and Framework Design

2.1. Core Concept of OBE

Outcome-Based Education, or OBE, has by now established itself as a standard reference point in emerging engineering education and in engineering accreditation more broadly. The premise is simple to state and hard to practice: we decide first what a graduate should be able to do, and only then work backward to the content, the activities, and the assessments that will get them there. This turns the usual teacher-led sequence on its head [2].

Spelled out, the idea rests on five commitments. First, success is possible for everyone, provided the instruction fits. The practical implication is that we design the course so that all students, not just the quickest, can reach the stated goals. Second, assessment has to be personal. Students arrive with different foundations and different capacities, and a single yardstick simply misses too much, which is why we need teaching and testing that adjust to the learner. Third, the bar is mastery. We expect students to understand a topic deeply enough that they can use it, not merely recognize it. Fourth comes accountability for results. The teaching team owns the outcomes and is expected to build a standing mechanism that keeps improving the course. Last, the real aim is competence for life beyond the classroom, including the habits of career growth and lifelong learning, rather than the recall of isolated facts [3].

On the practical side, OBE is usually carried by four habits. Every activity should keep a clear eye on the intended outcome. Students at different levels need different routes to the same destination. Expectation should be set high, using

staged and deliberately challenging tasks to draw out what students can do. And the whole talent goal is decomposed, layer by layer, into course objectives, content, projects, and the criteria we grade by.

2.2. Advantages of AI-Empowered Teaching

AI does not replace the teacher in this course, but it does fill several gaps that the traditional format leaves open. Our experience points to five of them, though they overlap more than any list suggests.

Take presentation first. Algorithms such as tree traversal, graph search, sorting, and recursion stop being dead text once they are shown as animation, short video, or a demonstration a student can poke at. The barrier to understanding drops because the student can finally see the thing move.

Tutoring is the next piece, and it is the one students notice most. An AI assistant that answers questions and debugs code is not bound by office hours or by the front of the room. A student stuck at midnight gets an explanation and a fix when the need is real, not two days later.

Then there is monitoring. The platform quietly logs almost everything: preview, what happens in class, the exercises done afterward, every code submission. Read together, those logs surface the misunderstandings the whole class shares and the particular holes a single student has. From there the system pushes micro-lectures, exercises, and cases pitched at the right level, which is what finally makes differentiated instruction something we can actually do, not just talk about.

The last two, practical training and assessment feedback, lean on simulation and online judging. They give students a cheap place to practice again and again, so hands-on work stretches across the entire course instead of one or two lab sessions. The payoff, modest but real, is that repetition stops being a luxury.

2.3. Overall Teaching Framework

Put the two strands together and a single framework emerges, one that loops back on itself through five layers: objectives, teaching, practice, assessment, and a final stage of improvement. We did not invent these layers so much as inherit them and then wire AI into each.

The loop begins where it should, with the talent-training program and what industry posts actually ask for. From there OBE's backward design sets the course aims along the familiar three lines of knowledge, ability, and the broader qualities we hope students carry away. AI's job is to supply the environment, the blend of online and face-to-face spaces where the content gets reorganized and where progressively harder projects teach students to put algorithms to use. All the while, an assessment that follows the student through the whole term feeds its findings, with the help of learning analytics, back into the plan, so the design keeps shifting in response to evidence.

In this arrangement the two parts divide the labor. OBE sets the direction of where the students are headed; AI supplies the means to move them there. Neither stands alone, and the combination is what lets the course build genuine, lasting ability rather than a passing familiarity with the material.

3. Current Course Status and Problems

3.1. Rigid Teaching Model and Dull Classroom Atmosphere

The dominant mode in the traditional classroom is the one-way lecture. Taught straight from the book and dense with logic, the sessions settle into a rhythm that is rigid and, frankly, dry. One format, delivered the same way, cannot hold attention for long. Interaction thins out, students slide into the role of passive listeners, and the habit of exploring on their own quietly fades. We have seen this happen term after term.

3.2. Abstract Knowledge Points and Difficult Algorithm Understanding

Trees, graphs, hash tables, and the harder sort routines all live at a level of abstraction that is hard to pin down on a page. A static diagram shows what a structure looks like, not what it does when it runs, and that gap is exactly where students get lost. They memorize the template instead of the mechanism, and the cost shows up later. Hand them the same problem in slightly different words, or ask them to adapt an algorithm to a new setting, and many simply will not know where to begin. That, more than any exam score, is the real measure of weak knowledge transfer.

3.3. Disconnection Between Theory and Practice and Weak Practical Ability

In its current form the course leans toward theory and away from practice. Lab hours are scarce, and most of the tasks students meet are variations on examples already in the book, not the integrated projects that software development or data processing would demand. The predictable result is that many students stop at the stage we might call 'I understand the code' and never reach the point of designing or debugging a program on their own. Theory, in other words, rarely hardens into a skill they can use. That sits awkwardly with the stated aims of application-oriented training [4].

3.4. Single Assessment System and Lack of Process Supervision

Assessment has settled into a fixed shape: a little regular homework, then one written final. What gets measured is mostly recall of the textbook, while the work that actually shows learning, the previewing, the participation, the coding, the teamwork, goes unrecorded [5]. A purely summative grade cannot speak to a student's fuller qualities, and it gives the teaching team no signal about where the course itself is breaking down. Without that signal, there is nothing to steer improvement by.

3.5. Lack of Layered Teaching and Difficulty in Tailored Instruction

A single classroom holds students whose programming background and logical reach differ widely, yet the traditional format proceeds at one pace with one body of content for all [6]. The weaker ones fall behind and then fall further, while the stronger ones wait, unchallenged, for the rest to catch up. The gap inside the room widens with every week that passes, and neither group is well served by the arrangement.

4. Construction of the OBE+AI Integrated Course Teaching System

4.1. Establishing Three-Dimensional Course Objectives Based on Backward Design

We set the course objectives the way OBE advises, working backward from the training program, the skills that industry posts call for, and the need to line up with neighboring courses [7]. Out of that come three strands of aim, knowledge, ability, and the broader qualities, and every activity in the course is meant to serve one or more of them.

On the knowledge side, students should leave able to handle both linear structures (lists, stacks, queues) and the nonlinear ones (trees, graphs), including how they are stored and why. They need the core logic behind sorting, searching, and hashing, they should be able to reason about time and space complexity, and they ought to know the code frameworks of the classic algorithms well enough to use them in a mainstream language.

Ability is where the course asks more of them. We want students who can think logically about an algorithm, take a problem apart, and then code, debug, and fix what they have built. Just as important is the judgment to pick the right structure or algorithm for a given engineering situation, plus the quieter habit of finding things out on their own and going on learning with the help of intelligent tools.

The quality strand is the easiest to undervalue and the hardest to assess. It asks for a serious, rigorous attitude toward the work, coding that follows standards, the give and take of teamwork and communication, and a readiness to push an algorithm toward something better. The thread running through all of it is an awareness that engineering itself calls for innovation.

4.2. Modular Reconstruction of Course Content and AI Visualization Optimization

Guided by the three strands of aim, we rebuilt the content as three modules that step up in difficulty, so that students at different levels each find a place. The first, which everyone must take, covers the basics: linear structures, elementary sorting, and searching. The second carries the real weight of the course, the binary trees, graphs, and the algorithms that give students trouble, and that is where most of the teaching effort goes. A third module reaches further, into greedy methods and dynamic programming, for those ready to stretch.

Visualization does the heaviest lifting where the ideas are most abstract. Our team built animation that shows an algorithm in motion and short videos that walk through code as it executes. When the class meets traversal, sorting, or path search, these resources lay out the whole sequence, every node visited, every swap, every change in a variable, so the mechanism stops being a mystery. Everything goes up on the online platform, which means students can meet it before class and return to it after [4].

4.3. AI-Supported Online-Offline Blended Teaching Model

We organized the term around the student, following OBE's lead, and split it into three phases that straddle the online and the face-to-face. There is work before the session, focused work during it, and consolidation once it ends.

Before class, the teacher puts up micro-lectures, preview tasks, and a few basic exercises. The AI system quietly

gathers what this produces, turns it into an analysis report, and pushes material matched to where each student actually stands, so differentiation starts before anyone walks into the room [7].

During the session, those reports let the teacher skip what the class has already got and spend the time on the difficulties most of them share. A typical class mixes visual demonstration, live coding, and discussion in groups. When a student is stuck, an AI coding helper or the question-answering tool can often unblock them on the spot, while the teacher moves through the room with more targeted help [8].

Afterward, the system sets homework scaled to how each student performed, ranging from routine exercises through variations on an algorithm to small development tasks. The question-answering tool stays on call at any hour and, usefully, gathers the harder questions it cannot answer and hands them to the teacher. Group projects round this out and give students a reason to apply the theory they have met.

4.4. Layered and Progressive Practical Teaching System

Competence, OBE reminds us, is built by doing, so we built a practice system in three tiers and backed it with online judging and simulation. The first tier is basic training, coding exercises pinned to a single point, linked-list operations or a tree traversal, where the AI grades automatically and walks the student through any error. Its only job is to make the foundations solid.

The second tier pulls several points together. Small engineering projects, a campus navigation tool or a grade-management system, are done in groups and force students to integrate what they know and to work with others.

The third is innovation training, deliberately open-ended. Students take on tasks like optimizing an algorithm or building on top of an open-source project, the kind of work that asks for original thinking and the higher-order skills the course is really after.

4.5. Full-Process Multi-Dimensional Assessment System

The old single-final model gives way to assessment that follows the student the whole way through and looks at more than one thing. The weights are set as follows: online learning accounts for 20 percent, classroom performance 15 percent, practical training 35 percent, and the final exam 30 percent.

Online learning is tallied by the platform itself, drawing on preview activity, time spent on micro-lectures, and the online quizzes. Classroom performance is judged by the teacher from interaction and live coding. Practical training splits its mark between AI scoring and the teacher's own read of code quality, project outcome, and teamwork. The final exam, for its part, goes after principles and the ability to apply them across the whole course.

None of this stays on paper. The AI pulls the full record together into a report on how far the course objectives have been met, and that report closes the loop: evaluate, analyze, improve, then begin again [9].

5. Expected Effects and Application Prospects of the Reform Scheme

The OBE-AI scheme was built for the specific troubles of this course and for the talent profile emerging engineering now asks for, and it is complete enough to run as written.

Where teaching is concerned, visualization takes the edge off the abstract material, and once learning analytics are paired with resources pitched to each student, the gaps inside a class start to close. Differentiated instruction becomes something a teacher can actually do, and students find a reason to study on their own. On the ability side, the step-by-step practice and the assessment that looks at more than one dimension pull students away from memorizing code and toward coding, solving, and applying in ways industry expects. For the course itself, the closed loop lets teachers keep tuning content and method, so quality improves term after term rather than standing still.

The design is not pinned to one course. Programming, operating systems, and algorithm analysis could absorb the same treatment, and as it spreads to other majors and other class sizes, the AI resources, the project difficulty, and the assessment weights can all be retuned as experience builds. That is how the marriage of idea and technology finally earns its place in practice.

6. Conclusion

Using Data Structures and Algorithms as the test bed, this study answers the call of emerging engineering education by joining OBE with AI to take on five stubborn problems: teaching that will not bend, material that stays abstract, theory left hanging above practice, a single summative grade, and no instruction shaped to the learner. The redesign touches the course in five places, its objectives, content, the way it is taught, the practice it asks for, and how it is assessed. OBE's backward design fixes where we are headed; AI carries the hard parts of delivery; and the blend of online and face-to-face work, step-by-step practice, and assessment that follows the student closes the loop and turns the course from teacher-led to learner-led. The scheme holds together and can be put to work, and we expect it to raise not only what students know but what they can do and how they think.

The study is not without its limits. The teaching resources we have built so far are mostly generic, and later rollout should bring in material tied to specific industries while the intelligent tools and the tutoring that adjusts to the student keep improving. Even so, the design gives a workable template for reforming engineering courses of the same family.

Acknowledgments

This research is funded by the project "Research on

Practical Teaching Reform of the Data Structures and Algorithms Course Integrating OBE Concept with AI Technology" (Project No. GKZLGC2025027). The authors thank all members of the teaching reform team for their valuable work in curriculum design, classroom implementation, and data collection. We also thank the students who took part in the reformed course and provided honest feedback; their involvement contributed greatly to the completion of this study.

References

- [1] Yin, X. Y., Ma, X., Hu, D., et al. (2025). Evaluation of architectural practice teaching based on OBE concept and emerging engineering background. *Shanxi Architecture*, 51(24), 185–189. (in Chinese)
- [2] Cao, P. (2025). Research on teaching reform of digital twin course based on outcome-based education. *Science and Education Guide*, (30), 66–68. (in Chinese)
- [3] Wu, C. H. (2026). Reform and practice of circuit analysis course teaching system based on integrating OBE concept and AI technology. (in Chinese)
- [4] Jin, S. G. (2025). Exploration of teaching reform of "Fundamentals of Data Structures" based on OBE concept in the context of artificial intelligence. *Social Science Forum*, 230–232. (in Chinese)
- [5] Tian, B., Guo, T. T., Zhu, L. Y., et al. (2026). Exploration of professional course teaching reform paths in application-oriented universities in the new era: Course practice of safety system engineering under OBE concept. *Safety*, 47(3), 77–82. (in Chinese)
- [6] Guo, Q., Ren, H., Zhang, Y. (2025). Construction and implementation path of core competency evaluation rubric system for undergraduates in application-oriented universities. *Modern Education Management*, (2), 99–108. (in Chinese)
- [7] Li, K. C., Wang, J. C., Liang, X. Y., et al. (2026). Discussion on curriculum reform of AI-empowered OBE concept: Taking Introduction to Oil and Gas Engineering as an example. *Inner Mongolia Petrochemical Industry*, (3), 68–71. (in Chinese)
- [8] Wu, X. P., He, P., Huang, C. (2026). Exploration of smart teaching mode of analog electronic technology course empowered by artificial intelligence under OBE concept. *China Educational Technology Equipment*, (8), 44–47. (in Chinese)
- [9] Peng, J. F., Cheng, P., et al. (2026). Practical teaching of comprehensive engineering training project based on OBE+CDIO concept. *Laboratory Science*, 29(1), 52–58. (in Chinese)