applied sciences

MDPI

Article

Efficiency-Centered Strategy Generation in Elementary Algorithm Education: An Instructional Design Framework Informed by Algorithm Engineering and Data-Driven Algorithm Design

Haeyoung Park [1] and Woochun Jun [2,*]

[1] Dandae Elementary School, Seongnam 13142, Republic of Korea; haepark.cs@gmail.com
[2] Department of Computer Education, Seoul National University of Education, Seoul 06639, Republic of Korea
* Correspondence: wocjun@snue.ac.kr; Tel.: +82-2-3475-2504

Abstract

Foundational algorithm instruction at the elementary level has centered on understanding control structures and reproducing given procedures; opportunities for learners to generate strategies and judge efficiency under varying data conditions remain limited. This article presents an instructional design framework for elementary algorithm education that adapts selected principles of Algorithm Engineering and Data-Driven Algorithm Design, with algorithmic complexity as a cross-cutting criterion, and reports how a formative expert evaluation informed its refinement. The framework was derived through a focused reconstruction of artifacts from a design and development study: principles were selected using three criteria, traceably mapped to design requirements, and the expert evaluation was re-examined. The reconstruction yielded five design requirements, six design principles, five operating principles, and a four-phase model: Recognizing Problems in Context, Exploring Analytical Solution Strategies, Implementing Condition-Responsive Procedures, and Reflective Evaluation and Optimization; formal proof and asymptotic analysis lie outside its scope. Nine of ten expert-rated items met the original study's content validity criterion, and expert feedback led to concrete learner-level activities, collaboration across all phases, and delayed presentation of canonical algorithms. The contribution is an explicit, traceable account of how professional practice principles were selected, bounded, adapted, and organized for efficiency-centered elementary algorithm instruction.

Keywords: algorithm education; algorithm efficiency; algorithm engineering; data-driven algorithm design; elementary education; formative expert evaluation; instructional design framework

Check for updates

Academic Editor: Vladimir Stantchev

Received: 19 August 2026
Revised: 28 September 2026
Accepted: 2 October 2026
Published: 5 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Algorithms are a core component of computational thinking [1–3], and the importance of developing foundations of algorithmic and procedural thinking from elementary education onward has long been emphasized [1,4,5]. In foundational algorithm instruction, however, understanding basic control structures such as sequence, selection, and repetition, along with explaining, reproducing, and representing given procedures, has been a central focus [1,4,6]. This focus helps learners understand basic algorithmic structures and develop procedural thinking. On its own, however, it does not ensure opportunities for learners to compare alternative strategies under different data conditions, judge efficiency, and

improve their procedures. At the elementary level, efficiency has been addressed only partially and has rarely been organized as learner-level content for comparing, selecting, and revising procedures under varying data conditions (Section 2.2) [7,8]. Accordingly, a systematic instructional design rationale is needed to place strategy generation and efficiency judgment at the center of instruction.

In professional practice, algorithm development extends beyond applying finished procedures; it is an iterative process that cycles through design, analysis, implementation, and experimental evaluation [9–11]. Algorithm selection and adjustment are likewise grounded in the characteristics of real data and observed performance [12,13]. Within this practice, algorithmic complexity serves as a disciplinary criterion for comparing how resource demands grow with input size, enabling efficiency differences among alternative procedures to be understood on a common scale [14,15]. The question, then, is how to translate these professional principles into activities for elementary learners who lack formal analytical skills, a question that the literature reviewed in Section 2 leaves largely open (Section 2.3). In this article, efficiency-centered refers not to formal complexity analysis but to instructional activities in which learners vary input conditions, such as the number of data items, and use observable execution evidence, such as step counts (the number of steps performed in a solution process) and execution time, to compare and improve their own strategies.

To address this question, this article has two objectives. The first is to present an instructional design framework for elementary algorithm education that translates the iterative practices of Algorithm Engineering and the data-grounded selection and adjustment principles of Data-Driven Algorithm Design into design requirements, design and operating principles, and a four-phase instructional model. The second is to report how a formative expert evaluation, conducted during the original design process, informed the refinement of that framework. The framework is derived from a focused reconstruction of documents generated during the design, development, and internal validation phase of the underlying design and development study [16] (Section 4.2). Its four phases are Recognizing Problems in Context, Exploring Analytical Solution Strategies, Implementing Condition-Responsive Procedures, and Reflective Evaluation and Optimization; formal proof and asymptotic analysis are placed outside its pedagogical scope.

The contribution of this article does not lie in claiming a wholly new theory or a first-ever integration of these professional practices with elementary education. Rather, it lies in making explicit and traceable how selected principles were identified, bounded in scope, pedagogically adapted, and organized into an instructional design framework. Specifically, the article contributes (1) a documented selection and scope-setting result that states which principles of Algorithm Engineering, Data-Driven Algorithm Design, and algorithmic complexity were adapted for elementary learners and which were excluded, with reasons (Section 5.1); (2) a traceable mapping that links five instructional design requirements to their disciplinary and professional practice foundations, learning-theoretical supports, documented design evidence, design and operating principles, and model phases, with the evidential status of each relation classified (Section 5.2); and (3) an account of how expert feedback was translated into specific revisions of the four-phase model, reported without claiming that the revised model was re-validated.

Accordingly, this article addresses three research questions.

RQ1. Which principles and practices identified in the disciplinary and professional practice literature, including Algorithm Engineering and Data-Driven Algorithm Design, met the selection criteria, and which elements were placed outside the pedagogical scope of the elementary framework?

RQ2. How were the principles from the selected foundations articulated as instructional design requirements, pedagogically adapted through selected learning-theoretical supports, and organized into design principles and a four-phase instructional model?

RQ3. How did the formative expert evaluation contribute to the refinement of the framework and the four-phase instructional model?

The remainder of the article reviews related work and identifies the research gap (Section 2), presents the conceptual background of the framework (Section 3) and the reconstruction methods (Section 4), reports the results for the three research questions (Section 5), and closes with the discussion, limitations, and conclusion (Sections 6 and 7).

2. Related Work

2.1. Foundational Algorithm Instruction

Algorithms occupy a central place in frameworks of computational thinking. They appear as a core concept in Brennan and Resnick's framework [2] and as one of the defined facets of computational thinking in Shute et al.'s review [3], while algorithmic thinking has been identified as a competence to be cultivated from the earliest grades [5,17]. Recent work has continued to refine what algorithmic thinking means for young learners, including its assessment in early childhood [18], and a recent reformulation of computational thinking for K–12 has positioned data, alongside algorithms and problems, as a central pillar [19].

Across much of this literature, the instructional treatment of algorithms at the elementary level has emphasized understanding basic control structures and representing or implementing procedures. A review of computational thinking education research characterized the field as concentrated on programming implementation and tool use rather than on the design decisions that precede implementation [4,6]. Elementary studies in Korea, where the original design study was conducted, show a parallel pattern: role play, unplugged card activities, educational programming languages, anchored problem situations, and flowchart or visualization materials have been used to make sorting and searching procedures accessible to young learners [20–24]. These studies demonstrate the feasibility of engaging elementary learners with standard procedures through activity-based representations and tools, and some extended the focus to problem-solving processes. However, efficiency was rarely made explicit as a central learning object, and many activities still treated established procedures primarily as objects of understanding, representation, or implementation. Learning outcomes were also commonly assessed through achievement or general thinking-skill measures rather than through the quality of the procedures learners produced.

2.2. Efficiency and Algorithm Comparison as Objects of Instruction

Efficiency and the comparison of alternative procedures have been studied mainly with older learners. A recent systematic review of algorithm design instruction found research on how students learn to design algorithms sparse even at the undergraduate level, with most topics covered by ten or fewer studies; among its takeaways, students found writing efficient and inefficient solutions and comparing their running times on real inputs more motivating than theoretical running-time analysis, and valued comparing alternative algorithms for the same problem [25–27]. Evidence from mathematics and computing education likewise indicates that comparing multiple solution methods supports procedural knowledge and the ability to select appropriate methods more effectively than presenting one method at a time [28,29], and a task-based study in secondary mathematics found that students could progress toward a general algorithm before the canonical method was introduced [30]. Taken together, these studies support two complementary instructional ideas relevant to the present framework: comparing alternative procedures using observable evi-

dence and allowing learners to develop strategies before canonical methods are introduced. However, these studies were conducted with secondary or postsecondary learners, and in the comparison studies cited here, the alternatives were generally instructor-provided rather than learner-generated.

At the elementary level, efficiency has entered instruction only partially. Moon proposed conceptual algorithm education with efficiency and optimization as design goals, but as a curricular direction rather than enacted activities [31]. Path selection in a routing task provided an indirect experience of efficiency [32], and an improvement stage was added after prediction and error correction in a proposed sequence of learning activities [33]. More recently, unplugged activities analyzing execution time were used with third graders [7], and complexity theory served as an analytic frame for classifying the algorithms that sixth graders constructed [8]. These studies extend elementary algorithm education toward efficiency, yet efficiency remained an indirect experience, a post hoc correction step, or a researcher's analytic lens; it was not organized as content that learners themselves use to compare, select, and revise procedures under varying data conditions.

2.3. Research Gap and Positioning of the Present Framework

Three gaps follow from this review, concerning content, instructional process, and design methodology. First, the literature reviewed here provides limited evidence that efficiency has been systematically developed as learner-level content in elementary algorithm education; learners have seldom been given explicit means to observe how the resource demands of a procedure change with input conditions and to use that evidence for judgment. Second, although comparison-based learning is supported by evidence, elementary designs in which learners generate their own strategies, compare them with peers' strategies and with canonical algorithms, and revise their procedures on the basis of execution evidence remain scarce. Third, the potential of professional practices such as Algorithm Engineering and Data-Driven Algorithm Design as sources of instructional design principles for elementary education remains underexplored, and the literature reviewed here provides limited evidence of an explicit, traceable account of how such practices can be selected, bounded, and pedagogically transformed into elementary-level design requirements and instructional procedures.

The framework presented in this article addresses these gaps: it operationalizes efficiency through observable execution evidence rather than formal notation, organizes comparison among learner, peer, and canonical strategies within a four-phase model with explicit re-entry pathways, and makes the translation from professional practice principles to design requirements and model phases traceable.

3. Conceptual Background

The framework rests on two categories of foundations. The first comprises disciplinary and professional practice foundations that provide principles for elementary algorithm education: two professional practice approaches—Algorithm Engineering and Data-Driven Algorithm Design—and algorithmic complexity as the disciplinary criterion that cuts across them. The second comprises pedagogical and learning-theoretical supports for translating these principles into activities for elementary learners: generative learning and the Deweyan tradition of inquiry and reflection. This section presents the core concepts of each foundation examined in the reconstruction and their potential implications for elementary algorithm education. Section 4 describes the selection procedures and criteria, and Section 5 presents the results of the selection and elementary-level adaptation.

3.1. Selected Disciplinary and Professional Practice Foundations

3.1.1. Algorithmic Complexity as a Cross-Cutting Disciplinary Criterion

Algorithmic complexity is a disciplinary criterion that describes the computational resources an algorithm requires as a function of input size [14,15]. Alternative algorithms for the same problem differ in how their operation counts grow as input size increases, and these growth patterns can be compared using common growth categories such as logarithmic, linear, and quadratic. In this framework, algorithmic complexity functions not as a third professional practice approach but as a cross-cutting disciplinary criterion for clarifying efficiency differences among procedures. The review focused not on formal analytical techniques themselves but on the underlying disciplinary insight: an algorithm's resource demands grow differently with input size, and alternative algorithms can be compared against a common criterion. The possibility that observed execution performance can also vary with input composition at the same input size connects this criterion to the empirical evaluation principles of Algorithm Engineering and Data-Driven Algorithm Design discussed below [10–13].

In plain terms, the criterion asks how much more work a procedure requires as the data grow: if one procedure's work grows roughly linearly with input size and another's roughly quadratically, moving from 10 to 100 items multiplies the first by ten and the second by a hundred. A second aspect is variation with input composition: for some procedures, such as insertion sort, nearly ordered data require far fewer steps than random data of the same size. The framework treats such input-dependent variation as something learners can observe through step counts and execution time, while asymptotic notation and formal analysis remain outside the elementary scope (Section 5.1).

3.1.2. Algorithm Engineering

Algorithm Engineering is an engineering approach that extends beyond treating algorithms solely as objects of mathematical analysis by situating them within an integrated development process encompassing real input data, implementation environments, performance measurement, and experimental evaluation [9,10,34]. Guided by testable hypotheses, development cycles through design, analysis, implementation, and experimental evaluation, with evaluation results feeding back into hypothesis revision and earlier phases [11]. This cycle is carried out not under idealized conditions but on realistic problem models and representative input conditions, with theoretical performance guarantees and empirical performance evidence operating in a complementary manner. For elementary algorithm education, this approach provides a professional practice model for reconceptualizing algorithm learning as a process of iterative design and empirical evaluation rather than the comprehension and reproduction of finished procedures.

Each phase has a distinct function [10,11]: design specifies the problem against a realistic model, states performance criteria, and formulates a testable hypothesis; analysis derives expected performance and, where possible, guarantees; implementation makes the design executable and measurable; experimental evaluation runs it on real or representative inputs and compares observed performance with the hypothesis, sending discrepancies back to earlier phases. In this perspective, specifying or proving an algorithm does not by itself complete the engineering process; its behavior on realistic inputs is also examined empirically. The framework draws on the cyclical structure with explicit return from evaluation and on the principle that judgments also draw on observed behavior on realistic inputs; the apparatus of professional implementation and benchmarking remains outside the scope (Section 5.1).

3.1.3. Data-Driven Algorithm Design

Data-Driven Algorithm Design grounds algorithm selection and adjustment not in designers' a priori preferences but in the data characteristics of the application domain and observed performance [12,13]. In this framework, it is positioned as a data-based decision-making principle that cuts across the Algorithm Engineering cycle from design to experimental evaluation. Its procedural flow comprises problem definition, collection of representative data, establishment of performance criteria, comparison of candidate procedures, and selection and revision based on execution results. For elementary algorithm education, this approach shifts judgments about algorithm quality away from authority or intuition and toward evidence grounded in data conditions and execution results.

Its starting observation is that the best algorithm often depends on which inputs actually occur, so that no single procedure is best for all cases [13]. Drawing on Balcan [12], the original study represented the approach as a six-stage flow, problem definition, data collection, performance criteria, algorithm selection and tuning, generalization analysis, and application, with return to earlier stages when the data change. The framework draws on problem definition, data collection, performance criteria, and comparison of candidate procedures as a procedural template, and on the principle that selection is grounded in evidence rather than fixed in advance; stages requiring statistical learning remain outside the scope (Section 5.1).

3.2. Selected Pedagogical and Learning-Theoretical Supports

The two pedagogical and learning-theoretical traditions below were integrated into the broader conceptual orientation of the original design study (see Section 4.2). This article distinguishes the pedagogical contribution of each tradition to translating professional practice principles into activities for elementary learners.

3.2.1. Generative Learning

Wittrock's generative learning theory explains learning not as the passive reception of information but as a process in which learners actively generate relations between new information and prior knowledge and experience, thereby constructing meaning [35,36]. This perspective was later developed into a broader account of active meaning construction and comprehension, emphasizing that learner-generated products such as summaries, explanations, and representations provide a central pathway to understanding [37,38]. In algorithm education, this tradition theoretically supports instructional designs in which learners generate and externalize their own solution strategies before canonical algorithms are introduced.

Two features of the theory bear on the framework. First, the unit of learning is the relation the learner constructs, not the information presented, so understanding is expected to differ depending on whether learners produce a solution or receive one. Second, the theory names activities such as summarizing, drawing, self-explaining, and teaching others through which learners externalize such relations [38]. The strategy sheets, flowcharts, and pseudocode produced in Phases 2 and 3 are generative products in this sense, and the delayed presentation of canonical algorithms is consistent with the first feature: presenting a canonical algorithm first could orient the activity toward reproduction rather than comparison with a learner-generated strategy (DR1; Section 5.2).

3.2.2. Deweyan Inquiry and Reflection

Dewey understood knowing not as the acquisition of fixed representations but as a process of reconstructing meaning by examining actions and their consequences in interaction with problem situations [39–41]. In this tradition, knowledge is reconstructed

through inquiry that begins in a problem situation, and the consequences of action serve as evidence for subsequent judgment, forming a reflective cycle that revises understanding. For algorithm education, this tradition offers two implications: comparison and evaluation activities that use procedural execution results as evidence for efficiency judgment, and a reflective cycle in which learners re-enter exploration and implementation phases after evaluation to revise their procedures.

Dewey described reflective thinking as moving from a felt difficulty to its definition as a problem, to a possible solution, to reasoning about its implications, and to testing by action, with the outcome revising the original idea [39]; in Logic, he characterized inquiry as the controlled transformation of an unsettled situation into a settled one, in which the consequences of acting on an idea are the evidence by which it is judged [40]. The framework takes from this tradition the status of execution results as evidence rather than a mere check of correctness, and the requirement that evaluation lead back into further exploration and implementation. These elements inform DR2, DR4, and DR5 and the model's re-entry pathways (Section 5.4).

Taken together, Algorithm Engineering provides the iterative development structure, Data-Driven Algorithm Design provides the principle of data-grounded selection and adjustment, and algorithmic complexity provides the disciplinary criterion for comparing procedures. Generative learning and Deweyan inquiry and reflection provide pedagogical support for translating these principles into learner activity through strategy generation, evidence-based comparison, and reflective revision. Section 4 describes how these elements were selected and mapped, and Section 5 reports the resulting framework.

4. Methods: Framework Design and Focused Reconstruction

Figure 1 summarizes the methodological process. The upper part shows the design, development, and internal validation phase of the original design and development study [16], whose documented artifacts constitute the source materials of this article. The lower part shows the four steps of the present focused reconstruction: identification of source materials (Section 4.1), theory-guided review and principle selection (Section 4.3; RQ1), traceable mapping (Section 4.4; RQ2), and re-examination of the formative expert evaluation (Section 4.5; RQ3). Section 4.2 specifies the focused reconstruction and transparency principles that govern these analytic steps. The documented artifacts of the original study's design phase, taken together, constitute the source materials from which the reconstruction proceeds.

4.1. Original Design Research Context and Source Materials

The framework reconstructed in this article was developed during the design, development, and internal validation phase of a design and development study on an algorithm education methodology for elementary learners [16]. This phase involved deriving design directions from a theoretical review, drafting an instructional model, conducting a formative evaluation with an expert panel and revising the draft, finalizing the model, and developing an eight-session instructional program for sixth-grade students. The materials analyzed in this article are the documented artifacts from this phase: the development directions and scope of the methodology (Table IV-1, pp. 115–117), the educational application of Algorithm Engineering and Data-Driven Algorithm Design and the resulting directions for algorithm education methodology design (Table II-14, p. 90), the theoretical foundations and design principles (Table IV-2, pp. 118–120), the draft and final versions of the model (Figures IV-1 to IV-4, pp. 122–134; Tables IV-6 and IV-7, pp. 135–137) and its operating principles (pp. 138–139), the expert evaluation materials (Tables III-1 and III-2, pp. 100–101; Tables IV-3 to IV-5, pp. 127–130), and the eight-session program documents (Table IV-8, pp. 142–143). Figure IV-4 is a system-level diagram that situates the instruc-

tional procedure within pre-lesson preparation and post-lesson extension. The program documents served as design evidence documenting how the framework was realized in classroom activities. Hereafter, tables and figures cited without further attribution refer to the original dissertation [16]; their page locations are given only in this inventory.

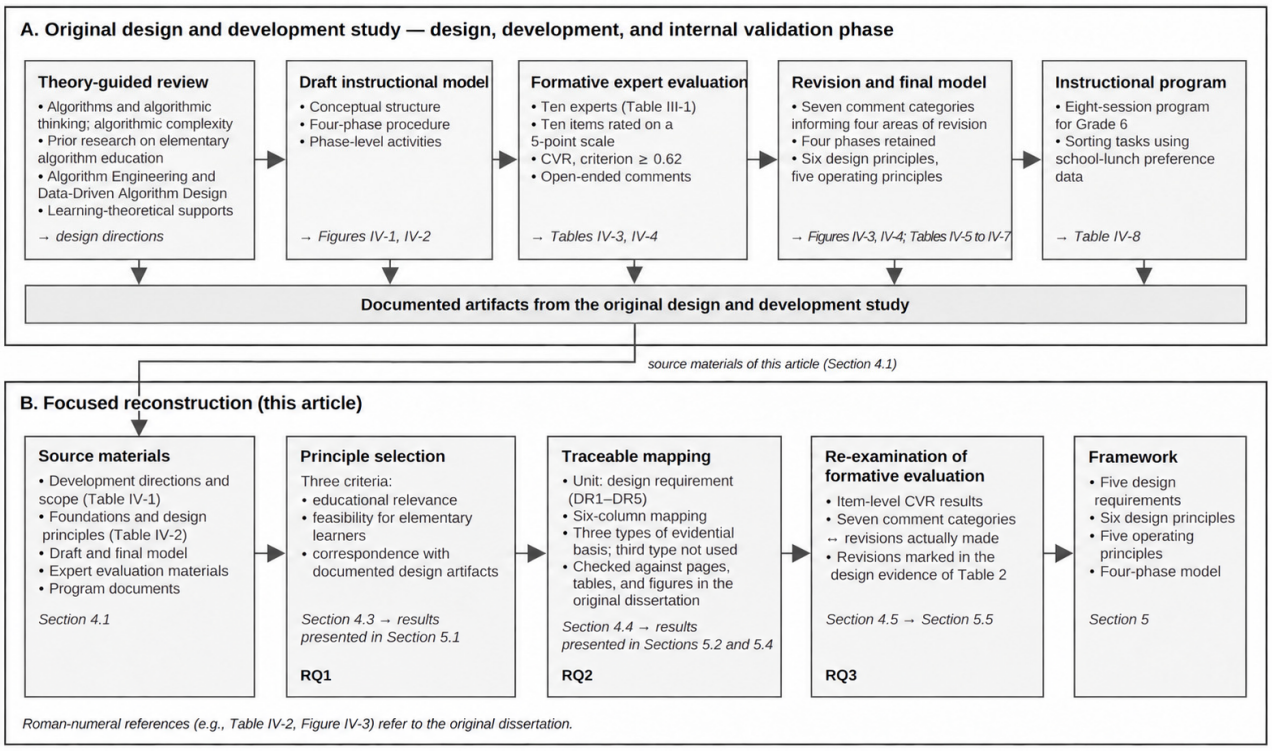


Figure 1. Methodological process of the study. (**A**) The design, development, and internal validation phase of the original design and development study [16], from theory-guided review to the eight-session instructional program. The documented artifacts of this phase constitute the source materials of the present article. (**B**) The focused reconstruction reported in this article, from source materials through principle selection and traceable mapping to the re-examination of the formative expert evaluation, with the research question addressed by each step. Roman numeral table and figure references (e.g., Table IV-2, Figure IV-3) refer to the original dissertation.

4.2. Focused Reconstruction and Transparency

The original design study was developed within a broader conceptual orientation termed generative epistemology, which drew on several epistemological and learning-theoretical traditions; in plain terms, it views learners as actively constructing and revising knowledge through interaction with problems, data, tools, and peers rather than receiving knowledge as a finished product. In the present article, this broader orientation is neither advanced as a new theory nor used as the organizing construct of the framework, for two reasons. First, it is broader than the specific disciplinary, professional practice, and pedagogical foundations isolated in the present reconstruction, and using it as the organizing construct would obscure the distinct contribution of each foundation that the traceable mapping is intended to make visible. Second, the contribution of this article lies in tracing how selected principles were bounded, adapted, and organized for elementary instruction; the broader theoretical justification for generative epistemology was developed in the original study and is not re-argued here. Instead, materials from the design, development, and internal validation phase were re-examined to identify and specify the contributions to instructional design of selected disciplinary and professional practice foundations, including algorithmic complexity, Algorithm Engineering, and Data-Driven Algorithm Design. The pedagogical supports presented in this article—generative learning

and the Deweyan tradition of inquiry and reflection—also operated within this broader orientation in the original design; the present reconstruction analytically distinguishes the contribution of each tradition. This focused reconstruction neither collects new empirical data nor alters the original design artifacts; it re-describes and organizes the relations between the documented artifacts and their evidential grounds so that those relations can be traced.

4.3. Theory-Guided Review and Principle Selection Procedure

The literature examined in this reconstruction was purposively selected from the sources used in the original study and limited to those directly related to the framework's focus. The original theoretical review comprised three strands: algorithms, algorithmic thinking, and elementary algorithm education; algorithmic complexity and professional approaches to algorithm development and selection, including Algorithm Engineering and Data-Driven Algorithm Design; and learning-theoretical research supporting the cyclical and interactive structure of the model. These strands informed four design directions used to draft the original model. Recent studies cited in Section 2 were added only to position the reconstructed framework within the current literature and were not part of the original design evidence.

For the professional practice foundations, the reconstruction retained sources that explicitly describe the process structures of Algorithm Engineering [9–11,34] and Data-Driven Algorithm Design [12,13], while algorithmic complexity was grounded in standard textbooks [14,15]. These sources were considered sufficient for this focused reconstruction because the pedagogical adaptation concerned the documented process structures and decision criteria of these approaches rather than their specific technical results. This was therefore a focused re-examination of the original design evidence, not a comprehensive or systematic literature review.

The principles and practice elements identified in this literature were repeatedly checked against the original design artifacts and assessed for inclusion using three criteria: educational relevance to the aims of elementary algorithm education, feasibility for elementary learners, and correspondence with documented design artifacts. Educational relevance was judged by whether an element contributed to strategy generation, comparison, evaluation, and improvement; feasibility by whether it could be translated into manipulable and observable activities without formal proof or advanced mathematical analysis; and correspondence by whether implementation evidence could be identified in the tables, figures, and lesson activities of the source materials. Principles meeting all three criteria were incorporated into the framework; elements not meeting one or more criteria were classified as outside the pedagogical scope of the elementary framework. The results are presented in Section 5.1.

4.4. Traceable Mapping Procedure

The chain from selected principles to design requirements, design and operating principles, and the four-phase model was organized into a six-column mapping, presented in Section 5.2. Each row uses one design requirement as the unit of analysis and connects it to the selected foundations, pedagogical supports, documented design evidence (cited by dissertation table and figure; page locations are listed in Section 4.1), corresponding design or operating principles, and location in the model. The evidential basis of each relation was classified into three types: relations explicitly documented in the original design materials; relations for which both the theory and design element were present in the source materials but the connection was implicit and analytically articulated in this reconstruction; and relations or theories introduced only in the present article. Theoretical

connections classified as the third type were not adopted as supporting grounds for the framework. This classification is also reflected in the wording used in the text: requirements directly articulated through review of a disciplinary or professional practice foundation are described as articulated from that foundation, whereas learning-theoretical connections that were implicit in the source materials are described as informed by the relevant theory. Requirements without an assigned external foundation are marked with an em dash (—) and retained on the basis of documented design evidence. Each mapping link was checked against specific pages, tables, and figures in the dissertation.

4.5. Formative Expert Evaluation

A panel of ten experts participated in the formative evaluation of the draft model (Table III-1). All panelists held doctorates, and their areas of expertise spanned computer education, informatics education, software education, and AI education. Nine were university professors and one was an elementary school vice principal; eight had ten or more years of teaching and research experience. The panel was composed to enable the draft model to be examined from both theoretical and classroom application perspectives.

The instrument comprised ten items (Table III-2): five on model characteristics (two on validity and one each on usefulness, universality, and clarity of expression) and five on phase-specific activity content (one per phase and one on the logical coherence of the procedure). All items were rated on a five-point Likert scale. The content validity ratio (CVR) was calculated by dichotomizing ratings of 4 and 5 as expert judgments of validity. For the ten-member panel, the original study applied Lawshe's critical CVR value of 0.62 [42]. Because attainable CVR values for a ten-member panel occur in increments of 0.20, meeting this criterion required at least nine of the ten experts to rate an item as valid (CVR = 0.80), which is also consistent with the exact binomial critical value reported by Ayre and Scally [43]. In the original study, items below the criterion were used to guide revisions rather than to justify removing the corresponding framework elements.

Open-ended comments were collected alongside the ratings to identify improvements that could not be captured by the quantitative results. The comments addressed the structural validity of the model, the logical connections among phases, the clarity of its details, and other suggestions for improvement. In the original study, these comments were organized into seven categories of suggested improvement: more specific lesson examples, level-appropriate treatment of complexity concepts, clearer representation of algorithmic procedures, more specific efficiency comparison and optimization activities, adjusted placement of collaborative activities, clearer procedures for comparison with canonical algorithms, and consideration of learners' starting points. For each category, the core concern expressed by the experts and the corresponding direction for model revision were documented (Table IV-4). The present reconstruction did not recode the original comments; it examined how each documented category corresponded to the revisions recorded in Table IV-5 and reports that correspondence in Section 5.5.

5. Results

5.1. Selected Disciplinary and Professional Practice Principles and Pedagogical Scope

The first research question concerns which principles and practices identified in the reviewed disciplinary and professional practice literature met the selection criteria and which elements were placed outside the pedagogical scope of the elementary framework. This section presents the results of selection and scope setting for Algorithm Engineering, Data-Driven Algorithm Design, and algorithmic complexity as a cross-cutting disciplinary criterion. Table 1 summarizes the overall structure.

Table 1. Principles selected from the reviewed disciplinary and professional practice literature and pedagogical scope setting.

Reviewed Disciplinary and Professional Practice Foundations	Selected Principles	Elementary-Level Adaptation	Elements Placed Outside the Pedagogical Scope
Algorithm Engineering [9–11,34]	Development structure cycling through design–analysis–implementation–experimental evaluation with feedback to earlier phases; empirical evaluation on realistic problem models and representative input conditions	Reorganized into a four-phase instructional procedure with explicit re-entry pathways from evaluation (Table II-14; Figure IV-3); realistic input converted into everyday data contexts such as school lunch menu preference data (Table IV-8)	Formal performance guarantees established by deductive proof; asymptotic analysis; development of reusable algorithm libraries; professional benchmarking systems based on automated measurement and repeated experiments—high formal and mathematical demands and limited feasibility at the elementary level
Data-Driven Algorithm Design [12,13]	Selection and adjustment of algorithms grounded in real data characteristics and observed performance rather than a priori preference; procedural flow of problem definition–data collection–performance criteria–comparison of candidate procedures	Data conditions made directly manipulable with data cards; performance criteria set as step counts and execution time, with comparison results across conditions visualized as graphs (Table IV-2; Table IV-5)	Machine-learning-based selection rules and predictive models; formal utility functions and parameter-tuning models; statistical learning procedures for generalization analysis—high formal and mathematical demands and limited feasibility at the elementary level
Algorithmic complexity as a cross-cutting disciplinary criterion [14,15]	Operation counts required by a procedure can vary with the size and composition of the input; the efficiency of alternative procedures can be compared against a common criterion	Operationalized at the elementary level so that learners vary input conditions such as the number of data items and compare observable execution indicators of step counts and execution time (Table IV-2; Table IV-6)	Asymptotic notation and formal complexity analysis—high formal and mathematical demands

Note. "Outside the pedagogical scope" refers to the scope setting of the elementary framework. Table and figure numbers refer to the original dissertation [16]; page locations are given in Section 4.1.

From Algorithm Engineering, the framework incorporated two elements: a cyclical development structure in which design, analysis, implementation, and experimental evaluation inform subsequent iterations, and the empirical evaluation of algorithms using realistic problem models and representative input conditions [9,10]. In the elementary framework, this cycle was reorganized into a four-phase instructional procedure with explicit re-entry pathways, while realistic input conditions were translated into everyday data contexts close to learners' lived experience. From Data-Driven Algorithm Design, the framework incorporated the principle that algorithm selection and adjustment should be grounded in the characteristics of real data and observed performance rather than a priori preference, together with a procedural flow of problem definition, data collection, establishment of performance criteria, and comparison of candidate procedures [12,13]. At the elementary level, these principles were translated into direct manipulation of data conditions through data cards, the use of step counts and execution time as performance criteria, and graphical visualization of comparison results across conditions. Formal proof

and asymptotic analysis, professional implementation and benchmarking systems, automated algorithm selection and tuning, and formal procedures for generalization analysis were placed outside the pedagogical scope because of their high formal and mathematical demands and limited feasibility at the elementary level.

Finally, algorithmic complexity was retained not as a third professional practice approach but as a cross-cutting disciplinary criterion for clarifying efficiency differences among procedures [14,15]. The framework operationalized this criterion through the relation between input conditions, such as the number of data items, and observable execution indicators, such as step counts and execution time, rather than through formal complexity analysis itself. This relation is not intended to replace formal complexity measures; it is a pedagogical representation for empirically exploring efficiency differences across input conditions at the elementary level. Asymptotic notation and formal complexity analysis were therefore placed outside the pedagogical scope. This placement defines the boundaries of the elementary framework; it does not judge the value of these elements in professional practice.

5.2. Articulated Design Requirements and Traceable Mapping

The second research question concerns how the selected disciplinary and professional practice principles were articulated as instructional design requirements, pedagogically adapted through selected learning-theoretical supports, and organized into design principles and the four-phase instructional model. The central product of this process is a set of five instructional design requirements (DR1–DR5). Table 2 maps each requirement to the foundation from which it was articulated, the corresponding design evidence, and the design and operating principles and model phases to which it leads. Its six columns present the articulated design requirement, selected disciplinary and professional practice foundation, pedagogical support, documented design evidence, corresponding design and operating principles, and location in the four-phase model.

Table 2. Traceable mapping of design requirements, selected foundations, documented design evidence, design and operating principles, and the four-phase model.

Articulated Design Requirement	Selected Disciplinary and Professional Practice Foundation	Pedagogical Support	Documented Design Evidence	Corresponding Design and Operating Principles	Location in the Four-Phase Model
DR1. Generation and externalization of learner strategies before canonical algorithms are introduced; canonical algorithms introduced as objects of comparison, not as starting points	Articulated from Algorithm Engineering (iterative design and analysis of alternative solution strategies)	Informed by generative learning	Operating principle "delayed presentation of canonical algorithms" (final model); Figure IV-3, Phase 4, "comparison of the execution results of student strategies and canonical algorithms"; clarification of the comparison flow (Tables IV-4 and IV-5, reflecting the formative evaluation)	Principle of procedural construction and representation; principle of context-based generative inquiry (Table IV-2)	Phase 2 → Phase 4
DR2. Comparison of alternative procedures under varying data conditions using observable execution evidence such as step counts and execution time	Articulated from Data-Driven Algorithm Design and from algorithmic complexity as a cross-cutting disciplinary criterion	Informed by Deweyan inquiry and reflection	Operating principle "data-based objective efficiency judgment" (final model); data-card activities, experience of step-count change, and visualized comparison of execution times (Tables IV-2, IV-5, IV-7)	Principle of data-based efficiency comparison (Table IV-2)	Phases 1, 2, and 4

Table 2. *Cont.*

Articulated Design Requirement	Selected Disciplinary and Professional Practice Foundation	Pedagogical Support	Documented Design Evidence	Corresponding Design and Operating Principles	Location in the Four-Phase Model
DR3. Explanation, comparison, and justification of strategies through structured collaborative interaction	—	—	Operating principle "collaborative comparison and explanation" (final model); peer strategy comparison in Phase 2 (Table IV-7); extension of collaborative activities across all phases after the formative expert evaluation (Table IV-4)	Does not correspond one-to-one to a separate design principle; "collaborative comparison and explanation" retained as an operating principle of the final model	All phases
DR4. Selection, revision, and improvement of procedures according to problem and data conditions	Articulated from Data-Driven Algorithm Design	Informed by Deweyan inquiry and reflection	Direction for the educational application of DAD in Table II-14; "judgment basis" in Table IV-6; design principle specification of data collection–comparison–analysis–selection and application activities (Table IV-2)	Principle of data-based judgment and application (Table IV-2)	Phases 2 and 4
DR5. Cyclical reflective structure in which learners re-enter the strategy exploration or procedure implementation phases after evaluation to improve their procedures	Articulated from Algorithm Engineering (design–analysis–implementation–experimental evaluation cycle)	Informed by Deweyan inquiry and reflection	Operating principle "cyclical feedback loop" (final model); explicit feedback pathways from Phase 4 to Phases 2 and 3 in Figure IV-3; Table II-14	Principle of the cyclical development process (Table IV-2)	Phase 4 → Phases 2 and 3

Note. The relations in the table do not imply that each instructional element was deductively derived from a single theory; they include relations explicitly documented in the original design process and relations analytically articulated in the present reconstruction. An em dash (—) indicates that no external foundation was assigned. Table and figure numbers refer to the original dissertation [16]; page locations are given in Section 4.1.

Following the criteria set out in Section 4.4, the relations in the mapping table are described as either explicitly documented in the original design process or analytically articulated in the present reconstruction from relations that existed implicitly in the design materials. An em dash (—) indicates that no external foundation was assigned to the requirement and that the requirement was retained on the basis of documented design evidence.

DR1 requires learners to generate and externalize their own solution strategies before canonical algorithms are presented, after which the canonical algorithms are introduced as objects of comparison; it was articulated from Algorithm Engineering and informed by generative learning. DR2 requires learners to compare alternative procedures under varying data conditions using observable execution evidence such as step counts and execution time; it was articulated from Data-Driven Algorithm Design together with algorithmic complexity as a cross-cutting disciplinary criterion. DR4 requires learners to select, revise, and improve procedures according to problem and data conditions, while DR5 specifies a cyclical reflective structure in which learners re-enter the strategy exploration or procedure implementation phases after evaluation. DR4 and DR5 were articulated from the condition-based selection and adjustment principle of Data-Driven Algorithm Design and the iterative development cycle of Algorithm Engineering, respectively; Deweyan inquiry and reflection informed the use of execution results as grounds for subsequent revision. DR3 requires

learners to explain and compare strategies and justify their choices through structured collaborative interaction. No separate external foundation was assigned to DR3, which was retained on the basis of documented design evidence specifying it as an operating principle of the final model. For DR1, DR2, and DR3, the evidence base includes revisions made in response to the formative expert evaluation (Section 5.5); Table 2 presents the requirement-level correspondences.

5.3. Design Principles

The latter part of the second research question concerns the system of design principles, which organizes the five design requirements into operational criteria for lesson design. Six design principles are documented in the original design materials (Table IV-2). The principle of context-based generative inquiry situates the start of learning in authentic problem contexts and data conditions and organizes inquiry, interaction, reflection, and feedback into a cycle. The principle of procedural construction and representation requires learners to construct solution strategies as ordered procedures and express them in flowcharts or pseudocode. The principle of the emergence of algorithmic thinking links conceptual understanding, design, representation, analysis, evaluation, and improvement within authentic problem solving. The principle of data-based efficiency comparison requires learners to vary input conditions, such as the number of data items, and empirically compare the resulting changes in step counts and execution time to judge efficiency. The principle of the cyclical development process connects problem recognition, reflective evaluation, and improvement within a sequential instructional process, while the principle of data-based judgment and application requires decisions to be grounded in data and execution results.

Rather than operating independently, the six principles are interconnected throughout learners' problem-solving processes. The original design materials summarize this interconnection along three axes. The first is authentic, context-based cyclical inquiry: learning begins in real problem situations and develops through a cycle of design, implementation, evaluation, and improvement. The second is procedural concretization and the emergence of algorithmic thinking: as learners' internal ideas are expressed as explicit procedures, their capacities for design, analysis, and evaluation emerge in an integrated way. The third is data-grounded efficiency judgment: algorithms should be evaluated through judgments grounded in data conditions and execution results rather than intuition or simple confirmation of outcomes.

The correspondence between each design requirement and the design and operating principles is shown in the fifth column of Table 2. DR3 does not correspond one-to-one to any of the six design principles; consistent with the documented design evidence, it was realized in the final model as the operating principle of collaborative comparison and explanation. The principle of the emergence of algorithmic thinking is integrative: rather than corresponding to any single requirement, it is realized through the joint operation of all five requirements, consistent with the interconnections assumed in the original design materials. This system of principles is operationalized in the four-phase instructional model presented in the next section.

5.4. The Four-Phase Instructional Model

The final product for the second research question is the four-phase instructional model, which organizes the five design requirements and six design principles into a single instructional procedure. Figure 2 presents the overall structure of the framework, from selected foundations through design requirements and principles to the model. The model comprises four phases (Table IV-6): Recognizing Problems in Context, Exploring Analytical

Solution Strategies, Implementing Condition-Responsive Procedures, and Reflective Evaluation and Optimization. These phases reorganize the design–analysis–implementation–experimental evaluation cycle of Algorithm Engineering into an elementary instructional procedure, with the core activities of each professional phase emphasized in the corresponding instructional phase (Figure IV-3). The decision-making principle of Data-Driven Algorithm Design operates across the model: learners identify problem and data conditions in Phase 1, explore alternative procedures and comparison criteria in Phase 2, and select, revise, and improve procedures according to execution results in Phase 4 (see Table 2). In the Phase 3 label, condition-responsive refers to constructing and adjusting procedures in response to changes in problem and data conditions. In Phase 4, optimization refers not to proving mathematical optimality but to comparing and improving procedures on the basis of observed execution evidence.

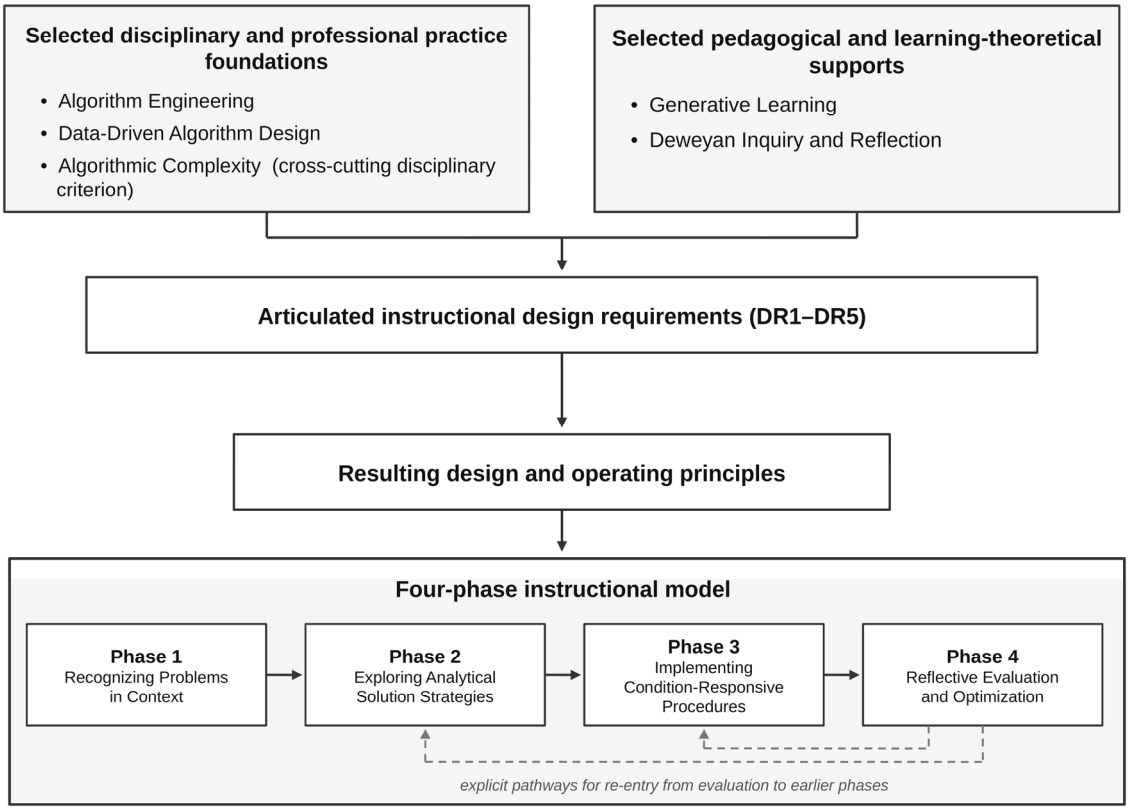


Figure 2. Instructional design framework for elementary algorithm education. Selected disciplinary and professional practice foundations—Algorithm Engineering, Data-Driven Algorithm Design, and algorithmic complexity as a cross-cutting disciplinary criterion—together with pedagogical and learning-theoretical supports inform five articulated instructional design requirements (DR1–DR5), six design principles, and five operating principles, which together support the four-phase instructional model. Dashed arrows indicate explicit pathways for re-entry from reflective evaluation and optimization to the strategy exploration and procedure implementation phases.

Learner activities and teacher supports are specified as phase-level implementation principles (Table IV-7). In Recognizing Problems in Context, learners analyze the constraints of the problem situation and the criteria for processing data and, through discussion, recognize the need for an algorithmic approach. The teacher presents the problem situation and provides prompts that promote exploration. In Exploring Analytical Solution Strategies, learners generate diverse solution strategies, compare them with peers' strategies, and examine step counts and processing approaches. The teacher guides the comparison criteria and points of attention for judging efficiency. In Implementing Condition-Responsive

Procedures, learners express and revise their strategies as executable procedures, such as flowcharts, pseudocode, or programs. The teacher provides feedback that helps them identify omissions, ambiguity, and inefficiency. In Reflective Evaluation and Optimization, learners use data size and execution results to compare and analyze procedural efficiency, interpret the sources of performance differences in relation to procedural structure, and improve their procedures.

The outputs generated in each phase function not merely as records of activity but as reflective tools for comparison, analysis, and procedural revision in subsequent phases. Strategy sheets are developed into flowcharts and pseudocode during implementation, while implementation results become execution time records and graph interpretation materials during evaluation. This chain of outputs links learners' design, implementation, evaluation, and improvement activities and supports the model's data-based judgment process in the classroom.

The four phases do not form a one-way linear structure. When inefficiency or errors are identified during evaluation, learners return to the strategy exploration or procedure implementation phases to revise and improve their procedures. These re-entry pathways are explicit in the final procedural schema (Figure IV-3), whose four-phase structure and re-entry pathways are presented in simplified form in Figure 2 of this article. This re-entry mechanism implements the cyclical reflective structure required by DR5 at the level of instructional procedure.

Finally, five operating principles for classroom implementation are documented in the final model: context-centered initiation, delayed presentation of canonical algorithms, data-based objective efficiency judgment, allowance of cyclical feedback loops, and collaborative comparison and explanation. These operating principles translate the design requirements into guidance for classroom implementation; their correspondences with individual requirements are documented in Table 2. The next section describes how the model's activities and representations were revised in response to the formative expert evaluation.

5.5. Formative Expert Evaluation and Framework Refinement

The third research question concerns how the formative expert evaluation contributed to refining the framework and the four-phase instructional model. Section 5.4 presented the structure of the refined final model; this section presents the expert judgments and resulting revision process. The evaluation focused on two draft artifacts. The draft conceptual structure (Figure IV-1) placed the original study's broader orientation at its base, positioned algorithms, algorithmic thinking, and algorithmic complexity as the central concept, core competence, and efficiency criterion respectively, and showed these being enacted through the Algorithm Engineering cycle and the Data-Driven Algorithm Design decision procedure. The draft instructional procedure (Figure IV-2) already comprised the four phases retained in the final model, each with a brief description of learner activity, and indicated a reflective feedback loop from execution results back to earlier phases, but without the later, more concrete specification of phase-level activities, learner outputs, or the placement of canonical algorithms and collaborative work. The original design materials state explicitly that, at the draft stage, whether the cyclical inquiry process and data-based judgment activities could be implemented validly and feasibly in actual classrooms remained an open question. The evaluation followed the procedure and criterion described in Section 4.5, and the full item-level results are provided in Table S1 in the Supplementary Materials.

Nine of the ten items met the criterion. Seven items—two on model validity, one each on usefulness and universality, two on Phase 1 and Phase 3 activities, and one on the logical coherence of the procedure—received a CVR of 1.00; the two items on Phase 2 and Phase 4 activities received a CVR of 0.80 (Table IV-3). The only item that did not reach the criterion

was "This instructional model presents an approach to algorithm instruction centered on algorithmic complexity in a readily understandable way" (CVR = 0.60), which was flagged for review. The original design materials interpreted this result not as evidence that the model's theoretical direction or phase structure was inappropriate, but as indicating a need to present more clearly how concepts such as algorithmic complexity, efficiency comparison, and optimization inquiry are realized as learner-level activities. This interpretation is consistent with the analysis of the open-ended comments below.

The open-ended comments fell into the seven categories of suggested improvement described in Section 4.5 (Table IV-4). The experts considered the model's theoretical direction and phase structure largely appropriate but indicated that the activities needed greater specificity to support classroom implementation. Across the seven categories, the common request was to translate abstract model elements into concrete activities and instructional supports that learners could enact. In particular, the operationalization of complexity and efficiency judgment was directly related to the below-criterion item on clarity of expression.

The original design materials summarize the resulting revisions in four areas (Table IV-5). First, complexity and efficiency judgment were translated into concrete learner-level activities. Data-card sorting and changes in step counts produced by varying the number of cards were incorporated into the problem recognition phase, and an activity in which learners examined step counts while observing peers' methods was added to the strategy exploration phase. Second, to move solution strategies beyond the level of ideas, activities involving flowchart and pseudocode representation, negotiation, and procedural revision were strengthened. Third, performance evaluation and optimization were specified as data-based judgment processes: changes in execution time across data sizes were visualized, and the results were used as evidence for algorithm selection. Fourth, collaborative activities, which had been concentrated in Phases 3 and 4 of the draft, were extended across all phases. The timing of canonical algorithm presentation was clarified so that learners constructed their own strategies in Phases 2 and 3 before comparing them with canonical algorithms in Phase 4, and intuitive foundational activities were added to Phase 1 to accommodate different learner starting points.

Throughout this refinement, the four-phase structure and model components were retained, and no element was removed. The changes focused on making the activities more specific and the representations clearer. Clarifying the timing of canonical algorithm presentation further specified DR1; the data-card activities, experience of changes in step counts, and visualization of execution time comparisons further specified DR2; and extending collaborative activities across all phases further specified DR3. These revisions are marked in the documented design evidence in Table 2. The revised model was not re-evaluated; the refinement should therefore be interpreted not as evidence that the identified issues were eliminated, but as a formative revision intended to improve clarity and suitability for elementary learners.

5.6. Summary of Results

The results can be summarized by research question. For RQ1, the reconstruction identified the principles that met the three selection criteria and the elements placed outside the pedagogical scope (Table 1). Algorithm Engineering contributed the iterative development cycle and empirical evaluation on realistic inputs; Data-Driven Algorithm Design contributed data-grounded selection and adjustment; and algorithmic complexity provided a criterion linking input conditions to observable execution indicators. Formal proof, asymptotic analysis, professional benchmarking, and automated selection were placed outside the elementary scope. For RQ2, these selected principles were articulated as five design

requirements (DR1–DR5), traceably linked to their foundations, pedagogical supports, and documented design evidence (Table 2), and organized into six design principles, five operating principles, and a four-phase model with explicit re-entry pathways (Figure 2). For RQ3, nine of ten expert-rated items met the original study's content validity criterion. The below-criterion item highlighted the need for clearer learner-level representation of complexity and efficiency judgment, while the seven categories of open-ended comments more broadly called for concrete activities and instructional supports. The resulting revisions further specified DR1–DR3 without removing any model element. Because the revised model was not re-evaluated, these findings describe formative refinement rather than validation of the final framework.

6. Discussion

This article presents an instructional design framework for elementary algorithm education through a focused reconstruction of documents from the underlying design and development study. Principles from Algorithm Engineering and Data-Driven Algorithm Design that met the selection criteria were articulated as five design requirements (DR1–DR5) and organized into six design principles and the four-phase instructional model. Algorithmic complexity was retained as a cross-cutting disciplinary criterion for comparing the efficiency of alternative procedures, while elements with high formal and mathematical demands, such as formal proof and asymptotic analysis, were placed outside the pedagogical scope (Sections 5.1–5.4). The formative expert evaluation identified one below-criterion item on clarity of expression and seven categories of open-ended comments; these findings guided the specification of activities and clarification of representations and were incorporated into the final model (Section 5.5). Because the revised model was not re-evaluated, the results indicate that expert judgment guided refinement of the activities and representations rather than establishing the validity of the final framework.

The core contribution of this article is a traceable mapping that connects selected disciplinary and professional practice foundations, pedagogical and learning-theoretical supports, and documented design evidence to design requirements, design and operating principles, and the four-phase instructional model. The contribution lies less in theoretical novelty than in making the processes of selection, scope setting, pedagogical adaptation, and organization explicit and traceable. In addition, distinguishing relations explicitly documented in the original design process from those implicit in the design materials, and marking requirements not attributed to external foundations with an em dash (Section 5.2), illustrate one way to transparently report how design elements relate to their theoretical and evidential grounds in instructional design frameworks.

The framework does not simply reduce the design–analysis–implementation–experimental evaluation cycle of Algorithm Engineering to an elementary procedure; it reorganizes the cycle into activities of strategy generation, comparison of execution evidence, and procedural revision that learners can perform [9–11]. Similarly, the selection and adjustment principles of Data-Driven Algorithm Design were translated not into formal learning models or automated tuning but into activities that use data conditions and observed execution results as grounds for judgment [12,13]. Delaying canonical algorithms to support learners' prior strategy generation is consistent with generative learning [35,36,38]. Using execution results as grounds for subsequent inquiry and revision translates the action–consequence–reconstruction cycle emphasized in Deweyan inquiry and reflection into an instructional procedure [39,40].

From a practical perspective, the framework offers several directions for designing elementary algorithm instruction: introduce canonical algorithms as objects of comparison with learner strategies rather than as starting points; have learners directly manipulate input conditions and compare alternatives using observable execution evidence; have learners

explain and justify their strategy choices to peers; and make re-entry pathways from evaluation to earlier phases explicit in the instructional procedure. These directions can be implemented through data contexts close to learners' lived experience and manipulable learning materials, using activities that do not require formal complexity analysis. From a theoretical perspective, the article provides an example of how professional practice can be translated educationally by selecting and reorganizing its underlying insights rather than simplifying its formal systems. These theoretical and practical implications concern the framework's design logic; its empirical effectiveness and applicability require separate classroom-based investigation.

7. Limitations and Conclusions

Four limitations delimit the claims of this article. First, the article reports the design logic and formative refinement of the framework but makes no claim about empirical effects on learning outcomes, which cannot be judged from the materials analyzed here. Second, the literature review was a focused re-examination based on the purposive selection of sources used in the original study, not a comprehensive or systematic review (Section 4.3). The selected foundations therefore should not be taken to encompass all theoretical and professional practice foundations relevant to elementary algorithm education; Section 2 positions the framework within related work, but an exhaustive comparison with existing instructional design models lies outside the scope of this article. Third, the framework's implementation evidence rests on documents from a single design case involving sorting algorithms and school lunch menu preference data; extending the framework to other algorithmic topics and data contexts will require additional design and evaluation. Fourth, the revised model was not re-evaluated during the formative expert evaluation. The reported refinement should therefore be interpreted not as demonstrating that the issues were eliminated, but as formative revision intended to improve clarity and suitability for elementary learners.

These limitations delimit the scope of the article's claims rather than undermine the claims made within that scope. Each concerns an inference the article does not make: empirical effects, exhaustive coverage of relevant foundations, generality across algorithmic topics and data contexts, or validation of the revised model. Within its stated scope—the framework's design logic, its traceable derivation from selected foundations and documented design evidence, and its formative refinement through expert judgment—the analysis is grounded in documented artifacts, reports the evidential status of the mapped relations, and makes the corresponding interpretive restrictions explicit.

The framework requires further examination through classroom-based studies that collect data on learner outcomes, learning processes, and implementation. Future research should examine how learners select different strategies in response to data conditions, explain the grounds for their choices using execution evidence, and revise a procedure or switch to another strategy in response to execution results when the framework is implemented in lessons. Design studies are also needed to examine how the design requirements and four-phase model can be implemented with algorithmic topics other than sorting and in different data contexts.

In conclusion, this article presents a framework that translates the iterative practices of Algorithm Engineering and the data-grounded selection and adjustment principles of Data-Driven Algorithm Design into instructional design for elementary algorithm education, with algorithmic complexity serving as a cross-cutting disciplinary criterion. The traceable mapping from five design requirements through design and operating principles to the four-phase instructional model provides a rationale and direction for complementing

instruction centered on canonical procedures and offers a starting point for subsequent classroom-based research.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/app16199860/s1, Table S1: Review items, content validity ratios, and decisions for the draft instructional model (N = 10).

Author Contributions: Conceptualization, H.P. and W.J.; methodology, H.P. and W.J.; investigation, H.P.; formal analysis, H.P.; data curation, H.P.; visualization, H.P.; writing—original draft preparation, H.P.; writing—review and editing, H.P. and W.J.; supervision, W.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The classroom implementation phase of the original design and development study, which involved sixth-grade students and is not analyzed in this article, was approved by the Institutional Review Board of Seoul National University of Education (approval number SNUEIRB-2025-007). The formative expert evaluation reported in this article was not included in that protocol. Separate ethical review was considered unnecessary because the panelists participated in their professional capacity as expert reviewers of a draft instructional model rather than as research participants from whom sensitive personal information was collected. Only limited professional-background information was collected, and the ratings and comments were reported without identifying individual panelists.

Informed Consent Statement: Informed consent was obtained from all panelists involved in the formative expert evaluation. The first page of the review form explained the purpose of the study, stated that the information and responses provided would be used only for research purposes and handled anonymously, and asked each panelist to indicate whether they consented to the collection of research-related information before completing the review; all ten panelists indicated their consent.

Data Availability Statement: The item-level results of the formative expert evaluation are provided in Table S1 of the Supplementary Materials. The source materials analyzed in this article are documented in the doctoral dissertation cited as [16].

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Grover, S.; Pea, R. Computational thinking in K–12: A review of the state of the field. *Educ. Res.* **2013**, *42*, 38–43. [CrossRef]
2. Brennan, K.; Resnick, M. New frameworks for studying and assessing the development of computational thinking. In Proceedings of the Annual Meeting of the American Educational Research Association, Vancouver, BC, Canada, 13–17 April 2012.
3. Shute, V.J.; Sun, C.; Asbell-Clarke, J. Demystifying computational thinking. *Educ. Res. Rev.* **2017**, *22*, 142–158. [CrossRef]
4. Lye, S.Y.; Koh, J.H.L. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Comput. Hum. Behav.* **2014**, *41*, 51–61. [CrossRef]
5. Hromkovič, J.; Kohn, T.; Komm, D.; Serafini, G. Algorithmic thinking from the start. *Bull. EATCS* **2017**, *121*, 132–139.
6. Kite, V.; Park, S.; Wiebe, E. The code-centric nature of computational thinking education: A review of trends and issues in computational thinking education research. *SAGE Open* **2021**, *11*, 21582440211016418. [CrossRef]
7. Kim, J.; Oh, M.; Yang, Y.; Kim, T.; Kim, J. The Effect of Algorithm Execution Time Analysis using Unplugged Program on the Computational Thinking of Third Grade Students. *J. Korean Assoc. Inf. Educ.* **2019**, *23*, 19–27. (In Korean) [CrossRef]
8. Kim, J. A Study on Algorithm Composition Patterns of Learners in Elementary Software Education. *J. Korean Assoc. Inf. Educ.* **2020**, *24*, 11–19. (In Korean) [CrossRef]
9. Mehlhorn, K.; Sanders, P. *Algorithms and Data Structures: The Basic Toolbox*; Springer: Berlin/Heidelberg, Germany, 2008.
10. Sanders, P. Algorithm engineering: An attempt at a definition. In *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*; Albers, S., Alt, H., Näher, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 321–340. [CrossRef]
11. Sanders, P. Algorithm engineering: An attempt at a definition using sorting as an example. In *2010 Proceedings of the Twelfth Workshop on Algorithm Engineering and Experiments (ALENEX)*; Blelloch, G., Halperin, D., Eds.; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2010; pp. 55–61. [CrossRef]

12. Balcan, M.-F. Data-driven algorithm design. *arXiv* **2020**, arXiv:2011.07177. [CrossRef]
13. Gupta, R.; Roughgarden, T. Data-driven algorithm design. *Commun. ACM* **2020**, *63*, 87–94. [CrossRef]
14. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 3rd ed.; MIT Press: Cambridge, MA, USA, 2009.
15. Sedgewick, R.; Wayne, K. *Algorithms*, 4th ed.; Addison-Wesley: Reading, MA, USA, 2011.
16. Park, H. A Study on the Development of an Instructional Methodology for Algorithm Education Based on Generative Epistemology. Doctoral Dissertation, Seoul National University of Education, Seoul, Republic of Korea, 2026.
17. Blannin, J.; Symons, D. Algorithmic thinking in primary schools. In *Encyclopedia of Education and Information Technologies*; Tatnall, A., Ed.; Springer: Cham, Switzerland, 2020; pp. 44–51.
18. Kanaki, K.; Kalogiannakis, M. Assessing algorithmic thinking skills in relation to age in early childhood STEM education. *Educ. Sci.* **2022**, *12*, 380. [CrossRef]
19. Palop, B.; Díaz, I.; Rodríguez-Muñiz, L.J.; Santaengracia, J.J. Redefining computational thinking: A holistic framework and its implications for K-12 education. *Educ. Inf. Technol.* **2025**, *30*, 13385–13410. [CrossRef]
20. Lim, H.; Kim, J.; Bae, J.; Baek, D.; Kim, H. A Instructional Learning of Sort Algorithm by Role-play for Computer Education in the Primary School. In Proceedings of the Korea Computer Congress, Daejeon, Republic of Korea, 23–24 April 2004; Volume 31, pp. 670–672. (In Korean)
21. Lee, Y.; Lee, Y. A Comparison of Teaching and Learning Method of Sorting Algorithm based on the Playing Activity and Animation. *J. Korean Assoc. Inf. Educ.* **2009**, *13*, 225–236. (In Korean)
22. Hyun, D.; Yang, Y.; Kim, E.; Kim, J. Research on Development and Application of Learning Program about Algorithm by Using Scratch Programming. *J. Korean Assoc. Inf. Educ.* **2011**, *15*, 387–397. (In Korean)
23. Choi, S.; Kim, Y. The Effect of Anchored Instruction on Elementary School Students' Problem-solving in Algorithm Learning. *J. Korean Assoc. Comput. Educ.* **2012**, *15*, 1–10. (In Korean) [CrossRef]
24. Han, B.; Ku, J.; Song, T. An Activity-based Instructional Design for Search Algorithm Expression of Elementary Students. *J. Korean Assoc. Inf. Educ.* **2016**, *20*, 161–170. (In Korean) [CrossRef]
25. Liu, J.; Poulsen, S.; Goodwin, E.; Chen, H.; Williams, G.; Gertner, Y.; Franklin, D. Teaching algorithm design: A literature review. *ACM Trans. Comput. Educ.* **2025**, *25*, 17. [CrossRef]
26. Silva, D.B.; de Lima Aguiar, R.; Dvconlo, D.S.; Silla, C.N. Recent studies about teaching algorithms (CS1) and data structures (CS2) for computer science students. In Proceedings of the 2019 IEEE Frontiers in Education Conference (FIE), Covington, KY, USA, 16–19 October 2019; pp. 1–8.
27. Gonda, D.; Ďuriš, V.; Tirpáková, A.; Pavlovičová, G. Teaching algorithms to develop the algorithmic thinking of informatics students. *Mathematics* **2022**, *10*, 3857. [CrossRef]
28. Rittle-Johnson, B.; Star, J.R. Does comparing solution methods facilitate conceptual and procedural knowledge? An experimental study on learning to solve equations. *J. Educ. Psychol.* **2007**, *99*, 561–574. [CrossRef]
29. Patitsas, E.; Craig, M.; Easterbrook, S. Comparing and contrasting different algorithms leads to increased student learning. In Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research (ICER '13), San Diego, CA, USA, 12–14 August 2013; pp. 145–152.
30. Lehmann, T.H. How current perspectives on algorithmic thinking can be applied to students' engagement in algorithmatizing tasks. *Math. Educ. Res. J.* **2024**, *36*, 609–643. [CrossRef]
31. Moon, G. On the Direction of the Computer Algorithm Education Based on Conceptual Algorithms. *J. Korean Assoc. Inf. Educ.* **2007**, *11*, 29–38. (In Korean)
32. Park, Y.; Kim, J.; Han, B. A Study on the Learnablity of Routing Algorithm in Elementary School Computer Education. *J. Korean Assoc. Inf. Educ.* **2007**, *11*, 267–279. (In Korean)
33. Kim, C. A Study on Algorithm Teaching and Learning Methods and Assessment for Elementary School Students. *J. Korean Assoc. Inf. Educ.* **2015**, *19*, 489–498. (In Korean) [CrossRef]
34. Demetrescu, C.; Finocchi, I.; Italiano, G.F. Algorithm engineering. *Bull. EATCS* **2003**, *79*, 48–63.
35. Wittrock, M.C. Learning as a generative process. *Educ. Psychol.* **1974**, *11*, 87–95. [CrossRef]
36. Wittrock, M.C. Generative learning processes of the brain. *Educ. Psychol.* **1992**, *27*, 531–541. [CrossRef]
37. Osborne, R.J.; Wittrock, M.C. Learning science: A generative process. *Sci. Educ.* **1983**, *67*, 489–508. [CrossRef]
38. Fiorella, L.; Mayer, R.E. Eight ways to promote generative learning. *Educ. Psychol. Rev.* **2016**, *28*, 717–741. [CrossRef]
39. Dewey, J. *How We Think: A Restatement of the Relation of Reflective Thinking to the Educative Process*; D. C. Heath: Lexington, MA, USA, 1933.
40. Dewey, J. *Logic: The Theory of Inquiry*; Henry Holt: New York, NY, USA, 1938.
41. McMurtry, A. Learning theory and interdisciplinarity: Reframing learning in terms of evolving coherences rather than "true" representations or correspondences. In *Handbook of Interdisciplinary Teaching and Administration*; Szostak, R., Ed.; Edward Elgar Publishing: Glos, UK, 2024; pp. 13–24.

42. Lawshe, C.H. A quantitative approach to content validity. *Pers. Psychol.* **1975**, *28*, 563–575. [CrossRef]
43. Ayre, C.; Scally, A.J. Critical values for Lawshe's content validity ratio: Revisiting the original methods of calculation. *Meas. Eval. Couns. Dev.* **2014**, *47*, 79–86. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.