Mehran University Research Journal of Engineering and Technology

https://doi.org/10.22581/muet1982.0792

2026, 45(3) 178-192

An Adaptive Gamification Optimization Algorithm for Analyzing Programming Learning Difficulties

Jairam Das Bather [a], Muhammad Saleem [b], Nisar Ahmed [c], Zahid Hussain Khaskheli [d*], Zoha Waheed [e], Ali Kolachi [d], Umer Ghaffar [f]

[a] *Department of Information Technology, Quaid-e-Awam University of Engineering, Science and Technology, Nawabshah 67480, Pakistan*

[b] *Department of Computer Science, Quaid-e-Awam University of Engineering, Science and Technology, Nawabshah 67480, Pakistan*

[c] *Faculty of Education Sciences and Technology (FEST), Universiti Teknologi Malaysia 81310, Malaysia*

[d] *Department of Software Engineering, Mehran University of Engineering and Technology, Jamshoro 76060, Pakistan*

[e] *Computational Science and Technology, National University of Sciences and Technology (NUST), Islamabad 4000, Pakistan*

[f] *Department of Statistical and Actuarial Sciences, University of the Punjab, Lahore 54590, Pakistan*

[*] Corresponding author: zahid.khaskheli@faculty.muet.edu.pk

Received: 17 Match 2026, Revised: 31 March 2026, Accepted: 22 June 2026, Published: 01 July 2026

KEYWORDS

Adaptive Gamification
Coding
Computer Education
Introductory Programming
PLDT
AGOA

ABSTRACT

Introductory programming courses worldwide continue to experience high failure an dropout rates, contributing to a persistent shortage of skilled software professionals This study develops and empirically evaluates an Adaptive Gamification Design Mode (AGDM) to address programming learning difficulties (PLDs). A three-phase mixed methods design was employed. Phase I used a PRISMA-guided systematic review c studies published from 2010 to 2025 (N = 112) to construct a multidimensiona Programming Learning Difficulty Taxonomy (PLDT) encompassing cognitive affective, and instructional challenges. In Phase II, the taxonomy was validated throug exploratory and confirmatory factor analyses of survey data from 842 undergraduat computer science and software engineering students at four public universities. Phas III comprised a 14-week quasi-experimental intervention involving a control group (= 142) and an adaptive-gamification group (n = 144). Cognitive load, syntax anxiety and self-efficacy deficit significantly predicted course failure. Compared wit traditional instruction, the AGDM environment produced higher academi performance and intrinsic motivation and reduced the dropout rate from 34% to 12% Structural equation modeling indicated that engagement and self-efficacy mediated th relationship between adaptive gamification and academic performance. The framewor is operationalized through the Adaptive Gamification Optimization Algorithm (AGOA), which dynamically adjusts task difficulty, feedback scaffolding, an motivational incentives according to each learner's PLDT profile. The study contribute a validated taxonomy and a scalable adaptive-gamification framework that can b integrated into computing curricula to support competence, retention, and personalize learning.

© Mehran University of Engineering and Technology 2026

1. Introduction

Introductory programming courses (CS1) are among the most difficult components of computer science and software engineering programs worldwide [1,2]. Despite curricular reforms and advances in computing education, failure rates of 30%-50% remain common, with averages near 40% in many settings [2, 3]. These outcomes threaten program sustainability and constrain the development of the future software and information-technology workforce [4]. Programming requires learners to understand syntax, reason abstractly, solve algorithmic problems, represent ideas symbolically, and debug across different environments [5,6]. For novices with limited prior coding or computational-problem-solving experience, these demands create substantial cognitive load [7].

Programming performance is not determined by cognition alone. Affective factors, including programming anxiety, low self-efficacy, fear of failure, and weak task-value perceptions, also influence outcomes [8,9]. Traditional lecture- and assessment-centered instruction often emphasizes task completion rather than the progressive development of foundational concepts [10]. Such approaches can intensify stress among students with limited programming backgrounds [11], while the interaction of cognitive overload and declining motivation contributes to attrition across computing programs [12].

Gamification—the application of game-design elements in non-game contexts—has attracted interest as a means of improving engagement and persistence [13]. Although empirical studies report increased participation and short-term motivation, many implementations depend on fixed systems of points, badges, and leaderboards [14]. These static designs rarely respond to learners' progress, emotional fluctuations, or performance patterns [15]. Consequently, they may generate initial interest without addressing the multidimensional difficulties involved in learning to program [16,17].

Two major gaps therefore remain. First, there is no sufficiently validated framework for classifying programming learning difficulties across cognitive, affective, pedagogical, and environmental dimensions [18-20]. Second, adaptive gamification remains underdeveloped theoretically and has rarely been examined through integrated, multi-phase research designs [21,22]. The psychological pathways through which adaptive gamification affects programming performance also remain incompletely understood [23].

To address these gaps, this study integrates a validated Programming Learning Difficulty Taxonomy (PLDT), a theory-driven Adaptive Gamification Design Model (AGDM), and the Adaptive Gamification Optimization Algorithm (AGOA). AGOA dynamically regulates task difficulty, feedback scaffolding, and motivational incentives according to learner profiles derived from PLDT. The multi-phase design combines a systematic literature review, large-scale factor analysis, a quasi-experimental intervention, and structural equation modeling [24], [25]. The study addresses three research questions: (1) Which latent multidimensional factors explain programming learning difficulties among undergraduate computer science and software engineering students? (2) Does taxonomy-based adaptive gamification significantly reduce predictors of failure in introductory programming courses? (3) Through which psychological processes, particularly engagement and self-efficacy, does adaptive gamification influence academic performance?

The study contributes by validating a multidimensional taxonomy, introducing a computationally supported adaptive-gamification framework, and empirically examining the pathways linking adaptive gamification with academic performance. Together, these contributions provide theoretical insight and practical guidance for improving retention and achievement in introductory programming education [26].

2. Literature Review

2.1 Persistent Challenges in Introductory Programming Education

Research in computing education consistently reports high failure and dropout rates in introductory programming courses across countries, institutions, and classroom settings [27], [28]. These findings indicate that programming difficulty is not a localized instructional problem but a widespread challenge in computer science education [29]. Novice learners must develop abstract reasoning and mentally simulate program execution, making the transition from basic computational exposure to formal program construction particularly demanding [30]. Although contemporary curricula increasingly emphasize problem solving, computational thinking, and software-development practices, student outcomes in introductory programming remain uneven. This pattern confirms that programming difficulty is multidimensional and requires structured theoretical models that account for interacting learning barriers [31,32].

© Mehran University of Engineering and Technology 2026

2.2 Cognitive Aspects of Programming Learning

Programming requires several demanding cognitive processes. Learners must understand execution logic, track variables and control flow, and apply abstract concepts in practical coding tasks [33]. These activities impose substantial intrinsic cognitive load, particularly on novices [34]. Cognitive Load Theory holds that working memory is limited; when instructional demands exceed this capacity, learning performance declines [35]. In programming courses, learners must simultaneously manage syntax, algorithm design, testing, and debugging, which can increase mental effort and reduce learning efficiency [36]. Recent computing-education research continues to identify cognitive overload as a major source of programming difficulty [37]. Novice programmers also struggle to transfer theoretical knowledge to authentic coding tasks, while debugging requires sustained logical reasoning and systematic analysis [38].

2.3 Affective and Motivational Factors in Programming Education

In addition to cognitive demands, affective variables strongly influence success in programming. Research in computer science education shows that programming anxiety, low confidence, fear of failure, and weak self-efficacy significantly affect engagement and persistence [39], [40]. Self-efficacy refers to a learner's belief in their ability to complete programming tasks successfully. Students with stronger self-efficacy are more willing to attempt difficult problems, persist during debugging, and invest time in practice. Conversely, low self-efficacy is associated with avoidance and reduced participation in programming activities [41], [42]. These findings indicate that effective interventions must address both cognitive and motivational dimensions of learning.

2.4 Teaching Methods in Programming Education

Conventional programming instruction often relies on lectures, exercises, and summative assessments [43]. Computing-education research increasingly recommends active-learning approaches, including project-based learning, peer instruction, and problem-based learning [44]-[46]. Technology-supported environments can extend these approaches by adapting to learners' needs and providing continuous interaction during coding practice [47]. However, active pedagogies do not inherently diagnose individual profiles of programming difficulty or adjust support accordingly, highlighting the need for structured adaptive mechanisms.

2.5 Gamification in Computing and Programming Education

Gamification applies game-design elements in non-game contexts to increase engagement and motivation [46]. In programming education, common elements include points, badges, leaderboards, progress tracking, and challenge-based activities [48]. Gamified coding environments can encourage learners to solve problems, complete exercises, and persist during debugging [49]. Nevertheless, many implementations use fixed reward structures that do not respond to individual performance, progress, or emotional state. Such systems may increase initial interest without sustaining long-term learning gains. The effectiveness of gamification therefore depends heavily on design quality [50]. Poorly designed systems may emphasize competition without improving conceptual understanding, which has led to growing interest in adaptive gamification.

2.6 Adaptive Learning and Adaptive Gamification

Adaptive learning systems modify instructional content according to a learner's skills, behavior, and performance. In programming education, they can calibrate task difficulty, provide individualized feedback, and recommend exercises based on demonstrated achievement. Adaptive gamification extends this approach by dynamically modifying challenge levels, feedback, and rewards using learner data. Matching instructional challenge to learner capability can support gradual skill acquisition, reduce frustration, and sustain motivation during difficult coding activities. Despite these advantages, limited research has combined taxonomy development, large-scale validation, experimental intervention, and structural modeling within a single adaptive-gamification framework for programming education.

2.7 Research Gaps in the Current Literature

The reviewed literature reveals four persistent gaps. First, no widely accepted multidimensional taxonomy integrates cognitive, affective, pedagogical, and environmental factors associated with programming learning difficulties. Second, most gamification studies emphasize short-term motivation rather than sustained performance and retention [51], [52]. Third, few studies combine taxonomy development and adaptive-system design with large-scale factor analysis and quasi-experimental validation [53]. Finally, the psychological mechanisms through which adaptive gamification influences programming performance remain insufficiently examined [54], [55].

3. Methodology

This study adopted a three-phase mixed-methods design integrating systematic synthesis, psychometric validation, and a quasi-experimental intervention. A sequential explanatory approach guided the research. Phase I developed the multidimensional Programming Learning Difficulty Taxonomy (PLDT) through systematic theoretical synthesis. Phase II empirically validated the taxonomy's latent structure. Phase III evaluated an adaptive gamification intervention operationalized through the Adaptive Gamification Optimization Algorithm (AGOA), which tailored task difficulty and game elements to learners' multidimensional difficulty profiles. Algorithm 1 summarizes the computation of the Programming Difficulty Risk Score (PDRS), task-suitability evaluation, gamification adjustment through the Gamification Motivation Index (GMI), and iterative learner-profile updates. The computational process is illustrated in Fig. 2.

3.1 Phase I: Systematic Review and Taxonomy Development

The systematic literature review followed the PRISMA guidelines [56] to identify and synthesize empirical evidence on programming learning difficulties published from 2010 to 2025. An a priori protocol was established to ensure transparency, reproducibility, and methodological rigor. The review addressed two guiding questions:

(1) What are the cognitive, affective, pedagogical, and environmental processes that cause learning difficulty in programming?

(2) How have these constructs been operationalized and empirically measured?

IEEE Xplore, Scopus, Web of Science, and ERIC were searched using Boolean combinations of terms related to programming difficulty, novice-programmer challenges, programming anxiety, introductory-programming failure, computational-thinking barriers, and gamification in programming education. The search was limited to peer-reviewed English-language journal and conference articles reporting empirical or systematic findings. The initial search yielded 386 records. After 84 duplicates were removed, 302 studies remained for title-and-abstract screening. Non-empirical studies (n = 71), studies outside programming contexts (n = 48), and opinion-based papers (n = 27) were excluded. Full-text assessment excluded studies without measurable constructs (n = 26) and studies conducted outside higher education (n = 18), leaving 112 studies for inclusion. The identification, screening, eligibility, and inclusion process is summarized in Table 1.

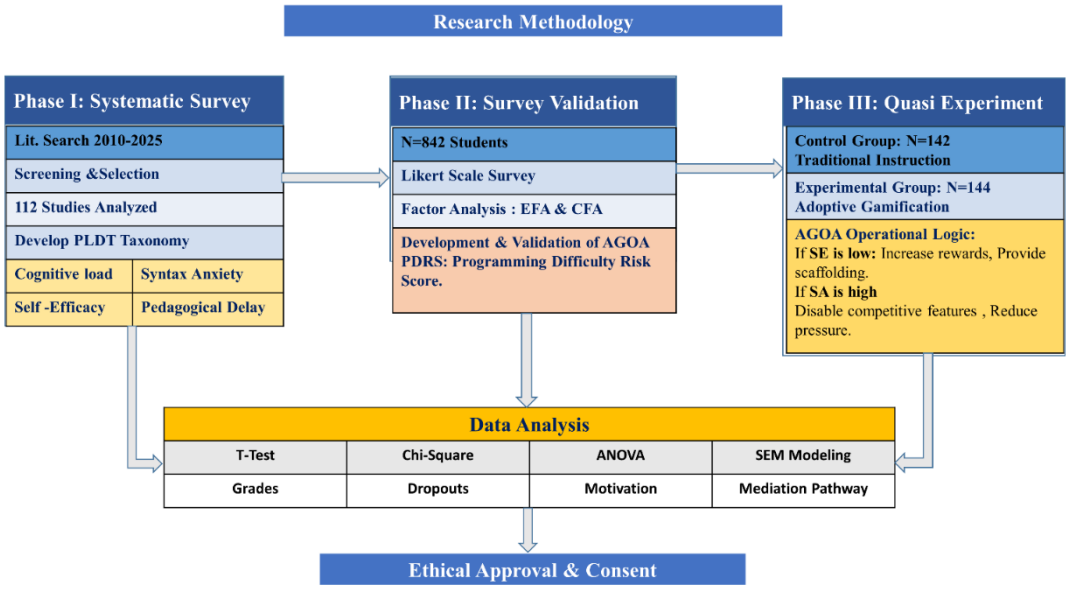


Fig. 1. Three-Phase Mixed-Methods Framework Showing Systematic Synthesis, Psychometric Validation, and Quasi-Experimental Intervention

Study characteristics, measurement instruments, statistical methods, and effect sizes were extracted. A hybrid inductive-deductive thematic-analysis procedure was applied [57]. Preliminary open coding identified 28 indicators of difficulty. Through iterative synthesis and construct clustering, these indicators were consolidated into 16 measurable indicators representing four higher-order latent constructs: Cognitive Load (CL), Syntax Anxiety (SA), Self-Efficacy Deficit (SE), and Pedagogical Delay (PD). Together, these constructs formed the initial Programming Learning Difficulty Taxonomy (PLDT).

A multidimensional construct profile was created to summarize the structural distribution of the

© Mehran University of Engineering and Technology 2026

synthesized domains, as shown in Fig. 3. The radar plot displays the relative contribution of each higher-order construct. Cognitive Load has the largest radial span, followed by Syntax Anxiety, while Self-Efficacy Deficit and Pedagogical Delay have smaller but distinct contributions. The visualization demonstrates that programming learning difficulty comprises differentiated yet related domains rather than a single dimension.

3.2 Phase II: Empirical Validation of the PLDT

3.2.1 Participants and Procedure

The taxonomy was validated using survey data from 842 undergraduates enrolled in introductory programming courses at four public universities. Stratified cluster sampling ensured representation from each institution. Eligibility was limited to first-year students taking their first programming course. The response rate was 78%, and the mean participant age was 19.8 years. Of the respondents, 62% were male, 38% were female, and approximately 71% reported no previous programming experience.

3.2.2 Instrumentation

Survey items were developed and refined using previously validated scales. Responses were recorded on a seven-point Likert scale ranging from strongly disagree to strongly agree. Before large-scale administration, the instrument was pilot-tested to assess clarity and content validity.

3.2.3 Analytical Strategy

Exploratory factor analysis used principal-axis factoring with Promax rotation. Sampling adequacy was high (KMO = 0.89), and Bartlett's test of sphericity was significant, $\chi^2(120) = 1456.7$, $p < 0.001$. Factors were retained based on eigenvalues greater than 1 and inspection of the scree plot, supporting a four-factor structure [58].

3.3 Phase III: Quasi-Experimental Intervention

3.3.1 Research Design

The 14-week quasi-experimental intervention included 286 students: 142 in the control group and 144 in the experimental group. Both groups were taught by instructors with comparable experience and used identical course materials, schedules, assessment formats, and laboratory facilities. These controls reduced potential confounding, allowing observed differences to be interpreted as effects associated with the intervention.

3.3.2 Intervention Conditions

Both groups followed the same curriculum and assessment structure. The experimental group used the AGDM framework, which incorporated adaptive difficulty calibration, mastery-based progression, progressive hint scaffolding, experience-point (XP) mechanics, and real-time formative feedback. After each exercise, the adaptive engine analyzed performance and recalibrated task complexity and feedback support to align challenge with the learner's progress.

3.3.3 Adaptive Gamification Optimization Algorithm (AGOA)

To operationalize the adaptive mechanisms embedded in AGDM, the Adaptive Gamification Optimization Algorithm (AGOA) was implemented as the learning environment's computational decision engine. AGOA dynamically regulates programming-task difficulty and motivational game elements according to multidimensional learner profiles derived from the PLDT. As summarized in Algorithm 1, the system computes the Programming Difficulty Risk Score (PDRS), evaluates task suitability, and adjusts gamification parameters using the Gamification Motivation Index (GMI).

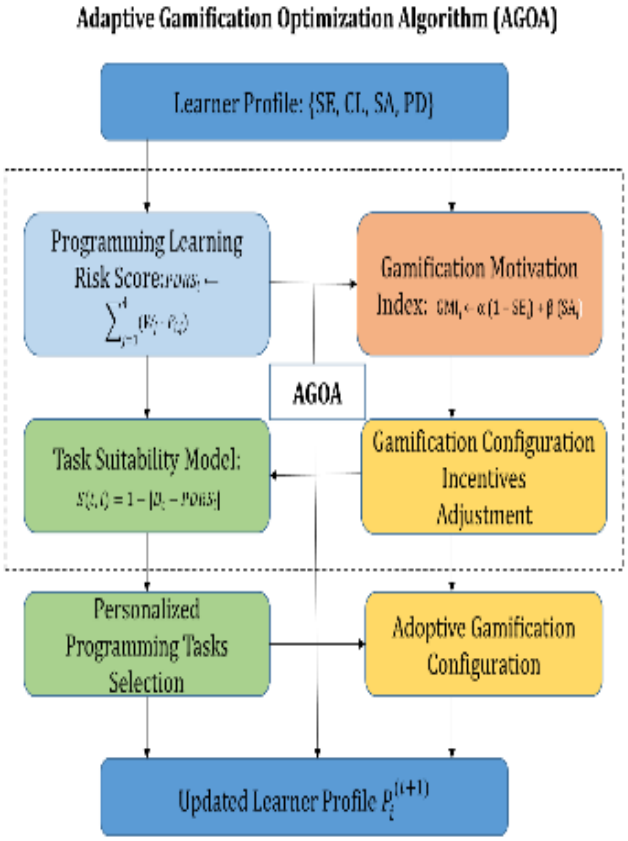


Fig. 2. Flow Diagram of the Adaptive Gamification Optimization Algorithm (AGOA)

© Mehran University of Engineering and Technology 2026

	Algorithm 1. Adaptive Gamification Optimization Algorithm (AGOA) ***Input:*** *Learner profile P_i; task set T; suitability threshold θ; weights W; maximum iterations T_{max}.* ***Output:*** *Personalized task Tadj; gamification configuration Γ; updated learner profile $P_i(t+1)$.* ***Initialization:*** *Normalize the learner profile to [0, 1] and set t = 0.*
Loop Condition	*While ($t < T_{max}$) and ($PDRS_i(t) > \varepsilon$)*
Step 1	*Compute: $PDRS_i = \Sigma_{j=1}^{4} W_j P_{i,j}$*
Step 2	*For each task $j \in T$, compute suitability: $S_{ij}(t) = 1 - \lvert PDRS_i(t) - D_j \rvert$*
Step 3	*Select the optimal task: $T_i^*(t) = \arg\max_j S_{ij}(t)$*
Step 4	*Compute the Gamification Motivation Index: $GMI_i = \alpha(1 - SE_i) + \beta(SA_i)$*
Step 5	*Configure the gamification policy $\Gamma(t)$ based on GMI_i.*
Step 6	*Deliver the selected task $T_i^*(t)$.*
Step 7	*Observe learner performance $Performance_i(t)$.*
Step 8	*Update the learner profile: $P_i(t + 1) = P_i(t) - \gamma \cdot Performance_i(t)$.*
Step 9	*Update the iteration counter: $t = t + 1$.*
Termination	*End while.*

To ensure computational reproducibility, AGOA is formally defined as a dynamic adaptive optimization model operating on a multidimensional learner profile. Each learner i at iteration t is represented by a standardized programming difficulty vector:

$$P_i(t) = \left[CL_{i(t)}, SA_{i(t)}, SE_{i(t)}, PD_{i(t)}\right] \quad \text{Eq. (1)}$$

- $CLi(t)$: Cognitive Load score
- $SAi(t)$: Syntax Anxiety score
- $SEi(t)$: Self-Efficacy Deficit score
- PDi(t): Pedagogical Delay score
- t: instructional iteration

All variables are normalized to the interval [0, 1]. Based on empirical effect sizes obtained in Phase II, a weighted model estimates the Programming Difficulty Risk Score (PDRS):

$$PDRS_i(t) = w_1 CL_i(t) + w_2 SA_i(t) + w_3 SE_i(t) + w_4 PD_i(t) \quad \text{we get:}$$

$$PDRS_i = \sum_{j=1}^{4} W_j P_{i,j}$$

Eq. (2)

Where: $w_1, w_2 \ldots w_4$ are standardized regression weights derived from Phase II empirical modeling.

The PDRS represents the predicted level of programming learning difficulty for learner i. AGOA uses this value to evaluate programming exercises in the task repository by comparing the learner's risk level with estimated task difficulty.

Task suitability is calculated as:

$$S_ij(t) = 1 - |PDRS_i(t) - D_j| \quad \text{Eq. (3)}$$

Task selection is defined as:

$$T_i^*(t) = \arg\max_j S_ij(t) \quad \text{Eq. (4)}$$

where D_t denotes the difficulty level of task t. Tasks that exceed a predefined suitability threshold are selected to create a personalized task set. Eq. (4) therefore selects tasks whose difficulty most closely matches the learner's current risk state.

To regulate motivational engagement, AGOA also computes a Gamification Motivation Index (GMI) from affective learner characteristics. Because self-efficacy and syntax anxiety are strongly associated with motivation and emotional responses during programming, these variables determine the intensity of gamification incentives:

$$GMI_i = \alpha(1 - SE_i) + \beta(SA_i) \quad \text{Eq. (5)}$$

© Mehran University of Engineering and Technology 2026

Subject to $\alpha + \beta = 1$. Where α and β represent weighting parameters controlling the influence of motivational variables. α controls motivational compensation; β controls anxiety buffering. Depending on the resulting index value, gamification elements such as rewards, feedback intensity, and competitive mechanics are dynamically configured to support learner motivation. Higher GMI values trigger supportive reinforcement mechanisms such as increased rewards and scaffolding, whereas lower values enable challenge-oriented competitive elements.

After each programming exercise, learner performance data are analyzed to update the learner profile vector:

$$P_i(t+1) = P_i(t) - \gamma \cdot Performance_i(t) \quad \text{Eq. (6)}$$

γ (gamma) denotes the learning-rate parameter.

Performance is scaled to the interval [0, 1], modeling a gradual reduction in programming difficulty over time. This iterative update enables continuous recalibration of task difficulty and gamification incentives across successive learning cycles, thereby aligning learner needs, instructional challenge, and motivational support throughout the semester.

Table 1
PRISMA Screening and Eligibility Process for Systematic Literature Selection

Phase	Action	No. of Records
Identification	Initial records identified from databases (IEEE Xplore, Scopus, WoS, ERIC)	386
Deduplication	Records removed before screening	84
Screening	Records screened by title and abstract	302
Exclusion 1	Records excluded (non-empirical, unrelated to programming, or theoretical only)	146
Eligibility	Full-text articles assessed for eligibility	156
Exclusion 2	Full-text articles excluded (no difficulty constructs: 26; non-higher-education context: 18)	44
Included	Total studies included in the synthesis	112

3.4 Ethical Considerations

Data collection began after approval was obtained from the relevant institutional review board. All participants provided informed consent, and the data were anonymized to protect privacy. Participation was voluntary and had no effect on academic grading.

4. Results and Analysis

This section presents the empirical findings from the three-phase research design used to evaluate programming learning difficulties and the effectiveness of the Adaptive Gamification Design Model (AGDM). The analyses cover the systematic-review outcomes, psychometric validation of the Programming Learning Difficulty Taxonomy (PLDT), predictive modeling of programming failure, and evaluation of the quasi-experimental intervention [59]. Statistical analyses were performed using IBM SPSS Statistics 26 and AMOS 24. Comparable statistical-modeling and performance-evaluation approaches are widely used in engineering and computing research to examine system behavior under varying operational conditions [60].

4.1 Phase I: Systematic Synthesis and Taxonomy Construction

4.1.1 PRISMA Screening Outcomes

The systematic search identified 386 records from IEEE Xplore, Scopus, Web of Science, and ERIC. After duplicate removal, 302 unique records remained for screening. Title-and-abstract review excluded 146 studies because they lacked an empirical design, were unrelated to programming, or presented theoretical commentary without measurable constructs. Of the 156 articles assessed in full text, 44 were excluded because they lacked operationalized difficulty constructs (n = 26) or were conducted outside higher education (n = 18). The final corpus comprised 112 empirical studies. Table 1 documents the PRISMA selection process and supports the transparency and reproducibility of the review. The included studies represented diverse geographical settings and

methodological approaches, strengthening the external relevance of the synthesized taxonomy.

4.1.2 Thematic Distribution of Programming Learning Difficulties

Open coding of the 112 studies initially identified 28 indicators of programming difficulty. Iterative axial coding consolidated these indicators into 16 measurable constructs grouped under four latent dimensions: Cognitive Load (CL), Syntax Anxiety (SA), Self-Efficacy Deficit (SE), and Pedagogical Delay (PD). The radar plot in Fig. 3 shows the relative variance contribution of each dimension. Cognitive Load accounted for the largest share (33.25%), followed by Syntax Anxiety (19.44%), Self-Efficacy Deficit (15.44%), and Pedagogical Delay (12.12%). This distribution confirms that programming learning difficulty is multidimensional rather than a single, unified problem.

Four factors with eigenvalues greater than 1.0 were retained, explaining 68.4% of the total variance in accordance with Kaiser's criterion. The eigenvalues were 5.32 for Cognitive Load, 3.11 for Syntax Anxiety, 2.47 for Self-Efficacy Deficit, and 1.94 for Pedagogical Delay. The scree plot showed a clear inflection after the fourth factor, supporting the four-factor solution.

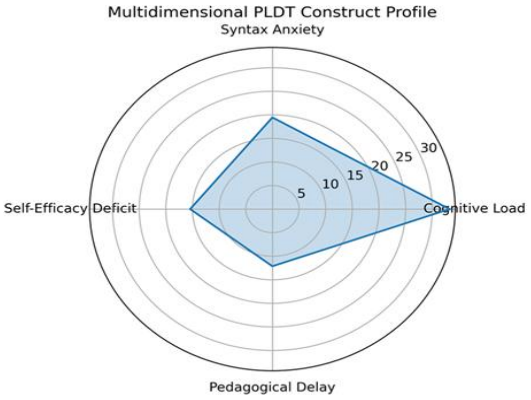


Fig. 3. Multidimensional PLDT Construct Profile

As shown in Fig. 4, eigenvalues declined sharply from the first to the fourth component and then leveled off. The fourth component formed the clear elbow of the scree plot. The first four factors had eigenvalues of 5.32, 3.11, 2.47, and 1.94, respectively, all exceeding 1. Subsequent factors had eigenvalues below 1 and contributed little additional explained variance. The scree plot therefore supports retention of the four conceptual dimensions: Cognitive Load, Syntax Anxiety, Self-Efficacy Deficit, and Pedagogical Delay. This four-factor solution provides a more parsimonious representation of programming learning difficulties than a larger factor structure.

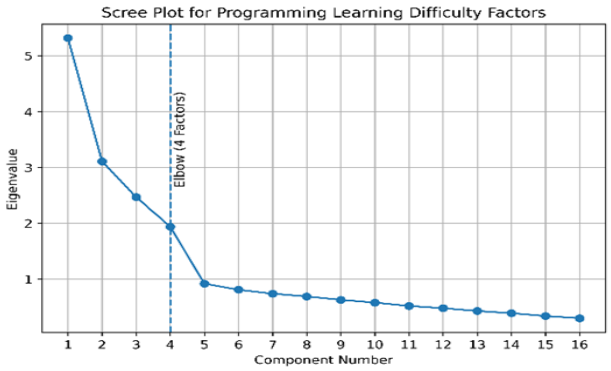


Fig. 4. Scree Plot of Eigenvalues from the Exploratory Factor Analysis

Table 2 presents factor loadings, cross-loadings, and communalities. All retained items loaded strongly on their intended constructs, with primary loadings ranging from 0.64 to 0.86. Cross-loadings were below 0.30, indicating satisfactory construct discrimination, while communalities ranged from 0.48 to 0.74. Cognitive Load explained the largest proportion of variance, consistent with theories emphasizing working-memory demands in programming. Syntax Anxiety emerged as a distinct affective challenge rather than merely an extension of cognitive load. Self-Efficacy Deficit showed the strongest item loadings, indicating high internal consistency among items measuring confidence in programming ability. Although Pedagogical Delay explained less variance, it remained a distinct factor representing instructional and contextual barriers.

These findings confirm that programming learning difficulties encompass interacting cognitive, affective, motivational, and instructional components rather than a single underlying cause.

4.1.3 Confirmatory Factor Analysis

Confirmatory factor analysis (CFA) using maximum-likelihood estimation was conducted to validate the measurement structure. The correlated four-factor model demonstrated good fit: CFI = 0.96, TLI = 0.95, RMSEA = 0.041 (90% CI [0.036, 0.046]), and SRMR = 0.038. Standardized factor loadings ranged from 0.68 to 0.87 and were statistically significant ($p < 0.001$). Reliability and convergent-validity statistics are presented in Table 3. Cronbach's alpha values ranged from 0.82 to 0.91, composite reliability values from 0.85 to 0.93, and average variance extracted values from 0.55 to 0.69. All exceeded recommended thresholds, while all HTMT ratios were below 0.85, supporting discriminant validity.

© Mehran University of Engineering and Technology 2026

Table 2
Rotated Factor Loadings and Communalities

Construct	Item	CL	SA	SE	PD	h^2
Cognitive Load (CL)	CL1	.81	.12	.08	.04	.69
	CL2	.77	.09	.11	.06	.63
	CL3	.74	.14	.05	.02	.59
	CL4	.69	.18	.07	.03	.55
Syntax Anxiety (SA)	SA1	.10	.83	.06	.05	.71
	SA2	.07	.79	.12	.04	.66
	SA3	.11	.76	.09	.08	.62
	SA4	.05	.72	.14	.06	.58
Self-Efficacy Deficit (SE)	SE1	.09	.07	.86	.05	.74
	SE2	.12	.10	.82	.03	.70
	SE3	.05	.13	.79	.04	.65
	SE4	.08	.06	.75	.07	.60
Pedagogical Delay (PD)	PD1	.04	.05	.09	.73	.57
	PD2	.07	.11	.06	.70	.54
	PD3	.10	.08	.04	.68	.51
	PD4	.06	.14	.07	.64	.48

Table 3
Construct Reliability and Validity

Construct	α	CR	AVE
Cognitive Load	.88	.90	.64
Syntax Anxiety	.86	.89	.61
Self-Efficacy Deficit	.91	.93	.69
Pedagogical Delay	.82	.85	.55

4.2 Predictive Validity of the PLDT Constructs

Multiple regression analysis examined the extent to which the four latent constructs predicted academic-failure risk. The model was statistically significant, $F(4, 837) = 185.32$, $p < 0.001$, and explained 47% of the variance in course failure ($R^2 = 0.47$). Self-Efficacy Deficit was the strongest predictor ($\beta = 0.62$, $p < 0.001$), followed by Cognitive Load ($\beta = 0.42$, $p < 0.001$), Syntax Anxiety ($\beta = 0.31$, $p < 0.001$), and Pedagogical Delay ($\beta = 0.19$, $p = 0.004$). This hierarchy indicates that perceived competence is a major vulnerability factor beyond cognitive demand alone. Variance-inflation factors were below 3.0, indicating no problematic multicollinearity. Fig. 5 compares the magnitude and statistical stability of the predictors.

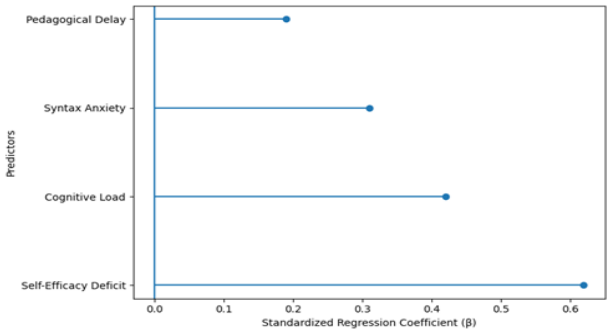


Fig. 5. Forest Plot of Predictive Effects

© Mehran University of Engineering and Technology 2026

4.3 Phase III: Quasi-Experimental Intervention Results

4.3.1 Academic Performance

An independent-samples t-test identified a statistically significant difference between instructional conditions. Students in the Adaptive Gamification Design Model (AGDM) group achieved higher final grades ($M = 83.0$, $SD = 7.9$) than those in the control group ($M = 64.3$, $SD = 8.7$), $t(284) = 19.42$, $p < 0.001$. The effect was large (Cohen's $d = 1.15$), and the mean difference of 18.7 percentage points was educationally substantial. As shown in Fig. 6, the AGDM score distribution shifted markedly to the right, with most scores concentrated in the low 80s, whereas control scores clustered in the mid-60s. The curves overlapped only slightly, mainly between scores of 70 and 75. The greater density of AGDM scores between 80 and 95 indicates broad improvement across the group rather than gains limited to a small subset of students.

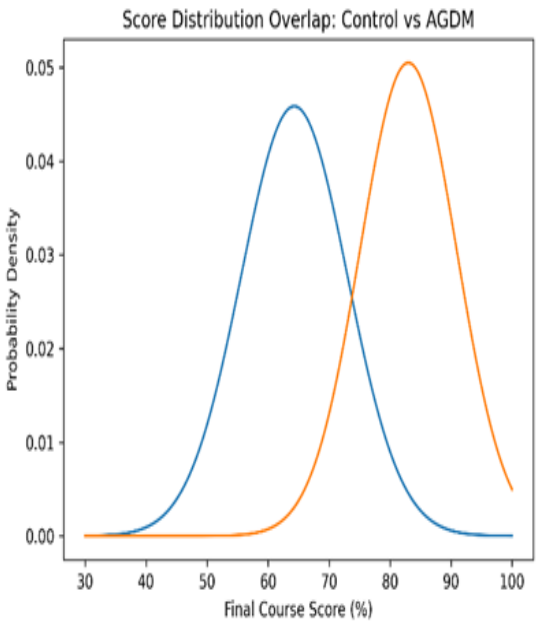


Fig. 6. Comparative Density Curves of Academic Performance for the AGDM and Control Groups

4.3.2 Dropout Reduction

The dropout rate was 12% in the AGDM group and 34% in the control group. A chi-square test confirmed that this difference was statistically significant, $\chi^2(1) = 18.74$, $p < 0.001$. The absolute risk reduction was 22 percentage points, equivalent to a relative reduction of 64.7%. From an institutional perspective, this represents a substantial improvement in retention. Survival analysis was conducted to examine retention dynamics throughout the intervention rather than only the final dropout proportion.

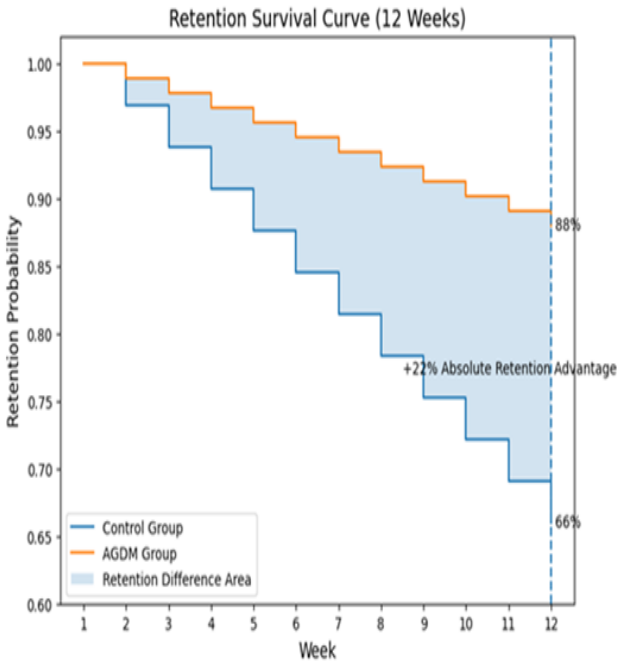


Fig. 7. Retention Survival Curves for the AGDM and Control Groups During the 12-Week Study

The retention curves in Fig. 7 declined gradually for both groups, but the AGDM group maintained a consistently higher retention probability. By Week 12, retention was 88% in the AGDM group and 66% in the control group, an absolute difference of 22 percentage points. The widening separation between the curves indicates that the retention benefit persisted over time. This pattern suggests that adaptive gamification produced a sustained protective effect against attrition rather than a short-lived motivational response.

4.3.3 Intrinsic Motivation

Analysis of variance showed a significant difference in intrinsic-motivation scores, $F(1, 284) = 116.32$, $p < 0.001$, with a large effect size ($\eta^2 = 0.29$). Students in the adaptive condition reported significantly greater perceived competence and enjoyment. This finding is consistent with motivational theory, which proposes that adaptive calibration strengthens psychological-need satisfaction.

4.3.4 Behavioral Engagement

Learning analytics showed that AGDM participants logged in 37% more frequently, completed 41% more tasks, made 52% more voluntary practice attempts, and spent 29% more time on task. These behavioral indicators demonstrate that adaptive gamification influenced not only self-reported perceptions but also observable learning behavior.

© Mehran University of Engineering and Technology 2026

5. Discussion

The findings show that programming learning difficulty is multidimensional rather than reducible to a single cognitive factor. Cognitive Load was the dominant dimension, consistent with Cognitive Load Theory and the substantial working-memory demands of programming. However, affective and motivational factors also contributed significantly, confirming that cognitive strain alone does not explain failure patterns. Syntax Anxiety emerged as a distinct affective state rather than a by-product of cognitive load. Self-Efficacy Deficit had the strongest regression coefficient ($\beta = 0.62$), identifying perceived competence as the principal vulnerability factor. Cognitive complexity may initiate difficulty, but declining self-efficacy can transform temporary challenge into disengagement and withdrawal. Although Pedagogical Delay explained less variance, it highlighted the structural importance of instructional pacing and timely feedback.

Overall, programming failure appears to result from interacting cognitive, affective, motivational, and pedagogical processes. The regression model explained 47% of the variance in course failure, representing substantial explanatory power in educational research. The predictive hierarchy—Self-Efficacy Deficit, Cognitive Load, Syntax Anxiety, and Pedagogical Delay—suggests a layered mechanism of failure development. Cognitive overload may initially produce confusion, whereas sustained disengagement is more closely associated with declining self-efficacy. Instructional design can reduce cognitive demand, but beliefs about competence may be harder to restore once learners perceive repeated failure. Motivation-focused interventions may therefore be more effective than content simplification alone.

5.1 Adaptive Gamification and the AGOA Mechanism

The quasi-experimental findings showed a large improvement in academic performance ($d = 1.15$) and a 64.7% relative reduction in dropout risk. Beyond the magnitude of these outcomes, the principal contribution lies in the mechanism through which the gains were achieved.

Within the Adaptive Gamification Design Model (AGDM), experience points, mastery-based progression, dynamic hints, and performance-based feedback created competence-supportive feedback loops. The Adaptive Gamification Optimization Algorithm (AGOA) operationalized this mechanism by analyzing learner-difficulty profiles derived from the Programming Learning Difficulty Taxonomy (PLDT) and dynamically adjusting task difficulty, feedback scaffolding, and motivational game elements. By combining risk estimation through the PDRS, task-suitability evaluation, and adaptive gamification policies, AGOA maintained an appropriate balance between challenge and learner competence. Unlike static systems that rely primarily on rewards, AGOA supports data-informed pedagogy that reduces repeated failure while sustaining engagement. This dynamic regulation appears central to the observed gains in motivation and academic performance.

5.2 Methodological Contributions

Methodologically, the study contributes a validated Programming Learning Difficulty Taxonomy (PLDT) and integrates systematic review, psychometric validation, predictive modeling, quasi-experimental testing, and mediation analysis within a single research framework. The factor solution explained 68.4% of the variance, while the predictive model explained 47% of course-failure variance, indicating strong empirical performance.

5.3 Implications, Limitations, and Future Research

In practical terms, AGDM reduced dropout by 22 percentage points, suggesting that psychologically grounded adaptive instruction may be more effective than content simplification alone. Effective implementation strategies include real-time scaffolding, competence-based task sequencing, failure-mitigating feedback, and engagement analytics. The study is limited by its quasi-experimental design, potential self-report bias, and restricted institutional context. Future research should assess the longitudinal stability of PLDT constructs, optimize AGOA through reinforcement learning, validate the framework across cultures and institutions, and incorporate real-time code analytics for more precise adaptation.

Conclusion

This study systematically examined the factors underlying programming learning difficulty and evaluated an adaptive instructional strategy for improving learning outcomes. The findings confirm that programming difficulty is multidimensional: Cognitive Load, Syntax Anxiety, Self-Efficacy Deficit, and Pedagogical Delay emerged as distinct but interrelated risk factors. Self-efficacy was the strongest predictor of course failure, indicating that learners' beliefs about their competence may influence success more strongly than conceptual complexity alone. The validated Programming Learning Difficulty Taxonomy (PLDT) provides a structured and measurable framework for diagnosing these barriers. Students exposed to the Adaptive

© Mehran University of Engineering and Technology 2026

Gamification Design Model (AGDM) achieved higher academic performance and intrinsic motivation and had lower dropout rates than students receiving traditional instruction. Structural equation modeling further indicated that self-efficacy and engagement mediated the relationship between adaptive gamification and academic achievement. The Adaptive Gamification Optimization Algorithm (AGOA) operationalizes the framework by dynamically adjusting task difficulty, feedback scaffolding, and motivational elements according to PLDT profiles. Although the quasi-experimental design limits causal inference and the results require replication across institutions, the study demonstrates that programming learning barriers can be addressed through theory-driven, adaptive, and psychologically supportive instruction. Future research should refine AGOA, test the cross-cultural validity of PLDT, and use real-time learning analytics to support proactive, data-driven, and individualized programming education.

References

[1] R. Mason, Simon, B. A. Becker, T. Crick, and J. H. Davenport, "A Global Survey of Introductory Programming Courses," in *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, Portland OR USA: ACM, Mar. 2024, pp. 799–805. doi: 10.1145/3626252.3630761.

[2] R. P. Medeiros, G. L. Ramalho, and T. P. Falcao, "A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education," *IEEE Trans. Educ.*, vol. 62, no. 2, pp. 77–90, May 2019, doi: 10.1109/TE.2018.2864133.

[3] Hafizul Fahri Hanafi *et al.*, "A Review of Learner's Model for Programming in Teaching and Learning," *J. Adv. Res. Appl. Sci. Eng. Technol.*, vol. 33, no. 3, pp. 169–184, Nov. 2023, doi: 10.37934/araset.33.3.169184.

[4] S. L. Marshall and A. A. Grooms, "Industry's push for computer science education: Is computer science really for all?," *Policy Futur. Educ.*, p. 14782103211045601, Jan. 2022, doi: 10.1177/14782103211045601.

[5] J. Tao, S. C. McClure, X. Zhang, M. Waqas, and X. Wen, "A scientific writing pedagogy and mixed methods assessment for engineering education using open-coding and multi-dimensional scaling," *Int. J. Technol. Des. Educ.*, vol. 30, no. 2, pp. 413–426, Apr. 2020, doi: 10.1007/s10798-019-09504-w.

[6] C. S. Cheah, "Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review," *Contemp. Educ. Technol.*, vol. 12, no. 2, p. ep272, May 2020, doi: 10.30935/cedtech/8247.

[7] H. G. Lapierre, P. Charland, and P.-M. Léger, "Looking 'Under the hood' of learning computer programming: the emotional and cognitive differences between novices and beginners," *Comput. Sci. Educ.*, vol. 34, no. 3, pp. 331–352, Jul. 2024, doi: 10.1080/08993408.2023.2214033.

[8] D. Huang, H. Wang, Y. Tang, H. Lei, and D. Koh, "Enhancing athlete performance under pressure: The role of attribution training in mitigating choking," *Frontiers in Psychology*, vol. 16, p. 1435374, 2025.

[9] A. Dirzyte, A. Vijaikis, A. Perminas, and R. Rimasiute-Knabikiene, "Associations between Depression, Anxiety, Fatigue, and Learning Motivating Factors in e-Learning-Based Computer Programming Education," *Int. J. Environ. Res. Public. Health*, vol. 18, no. 17, p. 9158, Aug. 2021, doi: 10.3390/ijerph18179158.

[10] C. S. Cheah, "Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review," *Contemp. Educ. Technol.*, vol. 12, no. 2, p. ep272, May 2020, doi: 10.30935/cedtech/8247.

[11] H. G. Lapierre, P. Charland, and P.-M. Léger, "Looking 'Under the hood' of learning computer programming: the emotional and cognitive differences between novices and beginners," *Comput. Sci. Educ.*, vol. 34, no. 3, pp. 331–352, Jul. 2024, doi: 10.1080/08993408.2023.2214033.

[12] C.-H. Su, "The effects of students' motivation, cognitive load and learning anxiety in gamification software engineering education: a structural equation modeling study," *Multimed. Tools Appl.*, vol. 75, no. 16, pp. 10013–10036, Aug. 2016, doi: 10.1007/s11042-015-2799-7.

[13] A. Khaldi, R. Bouzidi, and F. Nader, "Gamification of e-learning in higher education: a systematic literature review," *Smart Learn. Environ.*, vol. 10, no. 1, p. 10, Jan. 2023, doi: 10.1186/s40561-023-00227-z.

[14] L. Jaramillo-Mediavilla, A. Basantes-Andrade, M. Cabezas-González, and S. Casillas-Martín, "Impact of Gamification on Motivation and Academic Performance: A Systematic Review," *Educ. Sci.*, vol. 14, no. 6, p. 639, Jun. 2024, doi: 10.3390/educsci14060639.

© Mehran University of Engineering and Technology 2026

[15] R. Mellado and C. Cubillos, "Gamification improves learning: Experience in a training activity of computer programming in higher education," *J. Comput. Assist. Learn.*, vol. 40, no. 4, pp. 1959–1973, Aug. 2024, doi: 10.1111/jcal.13000.

[16] A. Khaleghi, Z. Aghaei, and M. A. Mahdavi, "A Gamification Framework for Cognitive Assessment and Cognitive Training: Qualitative Study," *JMIR Serious Games*, vol. 9, no. 2, p. e21900, May 2021, doi: 10.2196/21900.

[17] R. Mellado and C. Cubillos, "Gamification improves learning: Experience in a training activity of computer programming in higher education," *J. Comput. Assist. Learn.*, vol. 40, no. 4, pp. 1959–1973, Aug. 2024, doi: 10.1111/jcal.13000.

[18] K. Ishaq and A. Alvi, "Personalization, Cognition, and Gamification-based Programming Language Learning: A State-of-the-Art Systematic Literature Review," 2023, *arXiv*. doi: 10.48550/ARXIV.2309.12362.

[19] C. Baah, I. Govender, and P. R. Subramaniam, "Enhancing Learning Engagement: A Study on Gamification's Influence on Motivation and Cognitive Load," *Educ. Sci.*, vol. 14, no. 10, p. 1115, Oct. 2024, doi: 10.3390/educsci14101115.

[20] C.-H. Su, "The effects of students' motivation, cognitive load and learning anxiety in gamification software engineering education: a structural equation modeling study," *Multimed. Tools Appl.*, vol. 75, no. 16, pp. 10013–10036, Aug. 2016, doi: 10.1007/s11042-015-2799-7.

[21] L. Ding and H. Zhang, "The Gradual Cyclical Process in Adaptive Gamified Learning: Generative Mechanisms for Motivational Transformation, Cognitive Advancement, and Knowledge Construction Strategy," *Appl. Sci.*, vol. 15, no. 16, p. 9211, Aug. 2025, doi: 10.3390/app15169211.

[22] Z. Zhan, L. He, Y. Tong, X. Liang, S. Guo, and X. Lan, "The effectiveness of gamification in programming education: Evidence from a meta-analysis," *Comput. Educ. Artif. Intell.*, vol. 3, p. 100096, 2022, doi: 10.1016/j.caeai.2022.100096.

[23] J. Krath, L. Schürmann, and H. F. O. Von Korflesch, "Revealing the theoretical basis of gamification: A systematic review and analysis of theory in research on gamification, serious games and game-based learning," *Comput. Hum. Behav.*, vol. 125, p. 106963, Dec. 2021, doi: 10.1016/j.chb.2021.106963.

[24] S. Balaskas, I. Yfantidou, T. Nikolopoulos, and K. Komis, "The Psychology of EdTech Nudging: Persuasion, Cognitive Load, and Intrinsic Motivation," *Eur. J. Investig. Health Psychol. Educ.*, vol. 15, no. 9, p. 179, Sep. 2025, doi: 10.3390/ejihpe15090179.

[25] A. Mora, D. Riera, C. González, and J. Arnedo-Moreno, "Gamification: a systematic review of design frameworks," *J. Comput. High. Educ.*, vol. 29, no. 3, pp. 516–548, Dec. 2017, doi: 10.1007/s12528-017-9150-4.

[26] S. I. Malik and J. Coldwell-Neilson, "A model for teaching an introductory programming course using ADRI," *Educ. Inf. Technol.*, vol. 22, no. 3, pp. 1089–1120, May 2017, doi: 10.1007/s10639-016-9474-0.

[27] J. Figueiredo and F. Garcia-Penalvo, "Teaching and Learning Tools for Introductory Programming in University Courses," in *2021 International Symposium on Computers in Education (SIIE)*, Malaga, Spain: IEEE, Sep. 2021, pp. 1–6. doi: 10.1109/SIIE53363.2021.9583623.

[28] M. B. Garcia, "Profiling the skill mastery of introductory programming students: A cognitive diagnostic modeling approach," *Educ. Inf. Technol.*, vol. 30, no. 5, pp. 6455–6481, Apr. 2025, doi: 10.1007/s10639-024-13039-6.

[29] L. E. Margulieux, B. B. Morrison, and A. Decker, "Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples," *Int. J. STEM Educ.*, vol. 7, no. 1, p. 19, Dec. 2020, doi: 10.1186/s40594-020-00222-7.

[30] F. Hermans, "Hedy: A Gradual Language for Programming Education," in *Proceedings of the 2020 ACM Conference on International Computing Education Research*, Virtual Event New Zealand: ACM, Aug. 2020, pp. 259–270. doi: 10.1145/3372782.3406262.

[31] Y. Qian and J. Lehman, "Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review," *ACM Trans. Comput. Educ.*, vol. 18, no. 1, pp. 1–24, Mar. 2018, doi: 10.1145/3077618.

[32] R. Duran, A. Zavgorodniaia, and J. Sorva, "Cognitive Load Theory in Computing Education Research: A Review," *ACM Trans. Comput. Educ.*, vol. 22, no. 4, pp. 1–27, Dec. 2022, doi: 10.1145/3483843.

[33] J. Hanham, J. C. Castro-Alonso, and O. Chen, "Integrating cognitive load theory with other

© Mehran University of Engineering and Technology 2026

theories, within and beyond educational psychology," *Br. J. Educ. Psychol.*, vol. 93, no. S2, pp. 239–250, Aug. 2023, doi: 10.1111/bjep.12612.

[34] J. H. Berssanette and A. C. De Francisco, "Cognitive Load Theory in the Context of Teaching and Learning Computer Programming: A Systematic Literature Review," *IEEE Trans. Educ.*, vol. 65, no. 3, pp. 440–449, Aug. 2022, doi: 10.1109/TE.2021.3127215.

[35] J. Sweller, P. Ayres, and S. Kalyuga, *Cognitive Load Theory*. New York, NY: Springer New York, 2011. doi: 10.1007/978-1-4419-8126-4.

[36] R. Kumar, B. S. Chowdhry, and H. Kazi, "Identifying cognitive weaknesses in students' learning through Bloom's taxonomy," Journal of Information Communication Technologies and Robotic Applications, pp. 68–73, 2018.

[37] A. Emerson, A. Smith, F. J. Rodríguez, E. N. Wiebe, B. W. Mott, K. E. Boyer, "Cluster-Based Analysis of Novice Coding Misconceptions in Block-Based Programming," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*, New York, NY, USA: ACM, Mar. 2020..

[38] G.-Y. Lin, Y.-W. Liao, Z.-Y. Su, Y.-M. Wang, and Y.-S. Wang, "What drives undergraduates' effort and persistence in learning programming," *Educ. Inf. Technol.*, vol. 28, no. 10, pp. 12383–12406, Oct. 2023, doi: 10.1007/s10639-023-11670-3.

[39] Y. Pan and J. Harun, "Conquering Coding Fears: A Systematic Review of Programming Anxiety in Higher Education," *J. Inf. Technol. Educ. Res.*, vol. 24, p. 020, 2025, doi: 10.28945/5543.

[40] R. Thinakaran, S. Chuprat, V. Varadarajan, M. Yussubaliyeva, and M. Maqsood, "Self-efficacy as students' motivation factor in learning programming," *J. Infrastruct. Policy Dev.*, vol. 8, no. 16, p. 9252, Dec. 2024, doi: 10.24294/jipd9252.

[41] M. Sailer, J. U. Hense, S. K. Mayr, and H. Mandl, "How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction," *Comput. Hum. Behav.*, vol. 69, pp. 371–380, Apr. 2017, doi: 10.1016/j.chb.2016.12.033.

[42] S. R. Sobral, "Two different experiments on teaching how to program with active learning methodologies: a critical analysis," in *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*, Sevilla, Spain: IEEE, Jun. 2020, pp. 1–7. doi: 10.23919/CISTI49556.2020.9141115.

[43] I. Calderon, W. Silva, and E. Feitosa, "Active Learning Methodologies for Teaching Programming in Undergraduate Courses: A Systematic Mapping Study," *Inform. Educ.*, Sep. 2023, doi: 10.15388/infedu.2024.11.

[44] R. Kumar, N. Sarwar, K. Maheshwari, P. Lal, and K. Dev., "An EPIC technique for learning outcome assessment in OBE through Bloom's taxonomy," EAI Endorsed Transactions on Creative Technologies, vol. 6, no. 18, p. e3, 2019.

[45] C.-W. Tsai *et al.*, "The effects of online peer-facilitated learning and distributed pair programming on students' learning," *Comput. Educ.*, vol. 203, p. 104849, Oct. 2023, doi: 10.1016/j.compedu.2023.104849.

[46] X. Liu, "Implementation of Interactive Technology to Improve Students' Active Learning and Engagement in MATLAB Programming," in *2022 ASEE St. Lawrence Section Annual Conference Proceedings*, Syracuse University, New York: ASEE Conferences, Mar. 2022, p. 45414. doi: 10.18260/1-2--45414.

[47] F. Calles-Esteban, C. J. Hellín, A. Tayebi, H. Liu, M. López-Benítez, and J. Gómez, "Influence of Gamification on the Commitment of the Students of a Programming Course: A Case Study," *Appl. Sci.*, vol. 14, no. 8, p. 3475, Apr. 2024, doi: 10.3390/app14083475.

[48] A. N. Saleem, N. M. Noori, and F. Ozdamli, "Gamification Applications in E-learning: A Literature Review," *Technol. Knowl. Learn.*, vol. 27, no. 1, pp. 139–159, Mar. 2022, doi: 10.1007/s10758-020-09487-x.

[49] K. Ishaq *et al.*, "Level up your coding: a systematic review of personalized, cognitive, and gamified learning in programming education," *PeerJ Comput. Sci.*, vol. 10, p. e2310, Nov. 2024, doi: 10.7717/peerj-cs.2310.

[50] G. Çetin and S. Kuru Çetin, "GAMIFICATION IN COMPUTER ENGINEERING EDUCATION: A BIBLIOMETRIC REVIEW OF GLOBAL RESEARCH TRENDS," *Uluslar. Sürdürülebilir Mühendis. Ve Teknol. Derg.*, vol. 9, no. 1, pp. 159–173, Jun. 2025, doi: 10.62301/usmtd.1710461.

[51] M. E. Hangün and Y. K. Türel, "The Effects of Robot Programming on Mathematical Achievement, Mathematics Anxiety, and Programming Self-Efficacy," *Comput. Appl. Eng.*

Educ., vol. 33, no. 3, p. e70030, May 2025, doi: 10.1002/cae.70030.

[52] M. Moreno-Barahona, B. Fraijo-Sing, G. Fleury-Bahi, O. Navarro-Carrascal, and C. Tapia-Fonllem, "Conceptual Integration and Empirical Validation of a Unified Taxonomy: Quantitative Data Analysis for Virtual Learning Environments," *Front. Psychol.*, vol. 13, p. 814592, Apr. 2022, doi: 10.3389/fpsyg.2022.814592.

[53] "The Impact of Gamification Strategies on Motivation and Knowledge Retention in Learning Environments," *REST J. Bank. Account. Bus.*, vol. 1, no. 4, pp. 255–265, May 2025, doi: 10.46632/jbab/4/1/28.

[54] W. D. Huang, V. Loid, and J. S. Sung, "Reflecting on gamified learning in medical education: a systematic literature review grounded in the Structure of Observed Learning Outcomes (SOLO) taxonomy 2012—2022," *BMC Med. Educ.*, vol. 24, no. 1, p. 20, Jan. 2024, doi: 10.1186/s12909-023-04955-1.

[55] M. Ç. Uludağlı, "A Preliminary Analysis of Gamification on Assembly Language Programming Education," *Eur. Conf. Games Based Learn.*, vol. 19, no. 2, pp. 845–854, Sep. 2025, doi: 10.34190/ecgbl.19.2.3886.

[56] M. J. Page *et al.*, "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews," *BMJ*, p. n71, Mar. 2021, doi: 10.1136/bmj.n71.

[57] V. Braun and V. Clarke, "One size fits all? What counts as quality practice in (reflexive) thematic analysis?," *Qual. Res. Psychol.*, vol. 18, no. 3, pp. 328–352, Jul. 2021, doi: 10.1080/14780887.2020.1769238.

[58] L. Hu and P. M. Bentler, "Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives," *Struct. Equ. Model. Multidiscip. J.*, vol. 6, no. 1, pp. 1–55, Jan. 1999, doi: 10.1080/10705519909540118.

[59] C. Schulte, B. A. Becker, M. Divitini, and E. Barendsen, "[No title found]," in *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, Virtual Event Germany: ACM, Jun. 2021.

[60] *K. Mohammadani, R. A. Khan, K. A. Memon, N. N. Hussaini, J. H. Awan, and R. Kumar, "Stress-based performance analysis of AODV and DSDV routing protocols in MANET," Engineering Science and Technology International Research Journal, vol. 2, pp. 1–6, 2018.*