# Bridging the Digital Divide: A Global Framework for Transitioning from ICT Literacy to Authentic Computer Science Education

Asante Emmanuella Nana Akua

Xi'an Technological University, Xi'an, Shaanxi, China

University of Ghana, Legon, Accra, Ghana

asantenuella@gmail.com

**Abstract:** The persistent digital divide has evolved from a binary of access (haves vs. have-nots) to a nuanced spectrum of skills, ranging from basic Information and Communication Technology (ICT) literacy to the high-level abstraction and problem-solving inherent in Computer Science (CS) education. This review paper synthesizes three decades of research across educational technology, sociology, and computer science pedagogy to critically examine the gap between ICT literacy and authentic CS education. We propose a novel, globally applicable framework, the Competency Progression Ladder (CPL), which delineates five transitional stages: (1) Access & Basic Operation, (2) ICT Literacy, (3) Computational Thinking, (4) CS Fundamentals, and (5) Authentic CS Practice. The review identifies persistent barriers: infrastructural inequities (global South), teacher content knowledge gaps (universal), and sociocultural biases (gender and race). We analyze intervention models from six countries (Estonia, Rwanda, India, Brazil, USA, Finland) to extract scalable best practices. The paper concludes with policy recommendations for moving beyond digital consumerism toward digital authorship and innovation, arguing that authentic CS education is not a luxury but a civil right in the 21st-century knowledge economy.

**Keywords:** Digital divide; ICT literacy; computer science education; computational thinking; global education framework; K-12 pedagogy; educational equity

## 1. Introduction

The term "digital divide" was first popularized in the mid-1990s to describe the gap between those with access to computers and the internet and those without [1]. Three decades later, the discourse has shifted from first-level access (infrastructure) to second-level skills (usage) and third-level outcomes (economic and social benefits) [2, 3]. However, a critical oversight remains: the conflation of ICT literacy, the ability to use software, browse the web, and manage files, with Computer Science (CS) education, the systematic study of algorithms, data structures, programming, and computational problem-solving [4].

This conflation has led to policy failures. Many national curricula boast of "digital skills" while students merely learn to operate word processors or cloud tools [5]. Meanwhile, the demand for genuine CS competencies (e.g., logic, debugging, abstraction) has exploded, with global tech economies suffering from talent shortages [6]. The authentic challenge is no longer just providing access, but engineering a pedagogical bridge from consumption-oriented ICT skills to production-oriented CS thinking [7].

This review addresses three research questions:

1) What are the theoretical and practical distinctions between ICT literacy and authentic CS education?
2). What barriers (structural, pedagogical, sociocultural) prevent low- and middle-income countries (LMICs) and underserved communities from transitioning between these levels?
3) What evidence-based frameworks and interventions can facilitate a scalable global transition?

## 2. Methodology

This review followed a systematic literature synthesis approach [8]. We searched five electronic databases (ERIC, IEEE Xplore, Scopus, Web of Science, Google Scholar) for peer-reviewed articles

https://doi.org/10.58244/joerai.263890

published between 1995 and 2025 using the search string: ("digital divide" OR "ICT literacy" OR "computer science education" OR "computational thinking") AND ("K-12" OR "primary education" OR "secondary education") AND ("global" OR "developing countries" OR "equity"). Initial searches yielded 1,247 articles. After duplicate removal, title/abstract screening, and full-text review, 142 articles were included for final synthesis. Inclusion criteria required empirical data or theoretical frameworks addressing transitions between ICT and CS competencies in formal or informal educational settings.

## 3. Defining the Constructs: A Critical Distinction

A major contribution of this review is to clarify terminological confusion that pervades both policy and practice [4, 9].

### 3.1 ICT Literacy

ICT literacy is defined as the functional and critical skills to use digital tools for information retrieval, communication, and basic productivity (e.g., Microsoft Office, email, browser navigation) [10]. It is tool-dependent and consumer-oriented [11]. The International ICT Literacy Panel describes it as "the ability to use digital technology, communication tools, and/or networks to access, manage, integrate, evaluate, and create information" [12].

### 3.2 Authentic Computer Science Education

Authentic CS education is defined as the study of concepts that transcend specific tools: algorithms, data structures, computational logic, programming paradigms, debugging, and the design of comput ational artifacts [13]. Unlike ICT literacy, authentic CS emphasizes creation and abstraction rather t han procedural following [14].

### 3.3 The Gap: Computational Thinking as Bridge

Between these poles lies Computational Thinking (CT), a problem-solving methodology involving decomposition, pattern recognition, abstraction, and algorithm design [15]. However, CT is often taught as a standalone activity (e.g., unplugged exercises) without articulation to actual coding or CS principles, leading to what some researchers call "CT-washing" [16].Table 1 presents a systematic distinction between the two constructs.

Table 1 Distinguishing ICT Literacy from Authentic CS Education

| Dimension | ICT Literacy | Authentic CS Education |
|---|---|---|
| Core Activity | Using existing applications | Creating/computing with logic |
| Cognitive Demand | Procedural memory & search strategies | Abstraction, decomposition, systematic debugging |
| Typical Artifact | Document, spreadsheet, slide deck | Executable program, algorithm, data model |
| Error Handling | Troubleshooting UI/connectivity issues | Systematic debugging of logic errors |
| Tool Dependency | High (skills obsolesce with software) | Low (principles persist across paradigms) |
| End Goal | Digital task completion | Computational problem-solving |

## 4. Barriers to Transition: A Three-Dimensional Model

Our synthesis of 142 studies reveals three interacting categories of barriers that prevent progression from ICT literacy to authentic CS.

### 4.1 Structural and Infrastructural Barriers

In low- and middle-income countries (LMICs), unreliable electricity, low bandwidth, and severe device shortages persist [17, 18]. A UNESCO report found that 82% of sub-Saharan African schools lack basic internet access adequate for programming environments [19]. However, even in high-access regions, meaningful access is lacking: school computer labs are locked after hours, devices are underpowered for modern integrated development environments (IDEs), and technical support is absent [20]. The "second-level digital divide" [2] has now manifested as a computational divide: wealthier schools teach Python and data science while poorer schools teach keyboarding and Google Docs [21].

### 4.2 Pedagogical and Teacher Preparation Barriers

The single greatest bottleneck is teacher content knowledge [22]. Most K-12 teachers trained in ICT literacy (e.g., how to use a smartboard or learning management system) lack CS content knowledge (e.g., recursion, loops, data structures) [23]. A 2022 ACM survey found that only 35% of U.S. high schools offering CS had a teacher with a CS degree [24]. In LMICs, the figure is below 5% [25]. This leads to superficial instruction where spreadsheet exercises are mislabeled as "coding" [16].

### 4.3 Sociocultural and Affective Barriers

Gender stereotypes remain powerful: girls are subtly steered toward ICT literacy (digital communication, design tools) while boys are encouraged toward CS (programming, robotics) [26, 27]. Longitudinal studies show this gap widens between ages 12-15 [28]. Additionally, a "culture of fear" around mathematics and logic alienates first-generation learners [29]. In many countries, CS is perceived as elitist or irrelevant to local livelihoods, reducing motivation among rural and minority students [30].

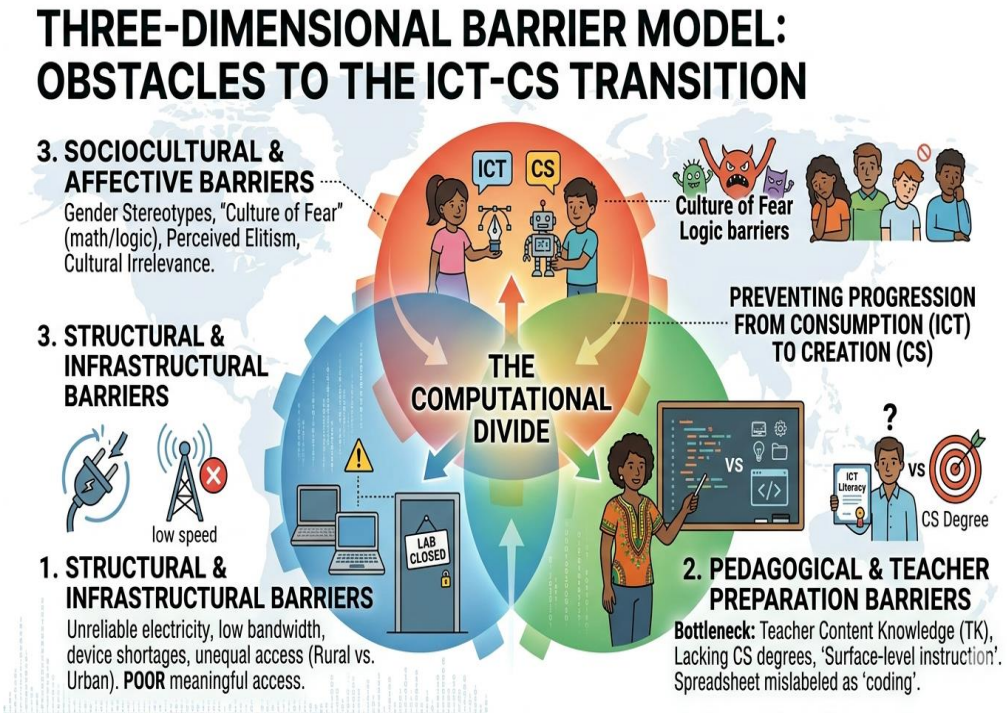



Figure 1. three-dimensional barrier model

## 5. Global Interventions: A Comparative Analysis

We analyzed six national/regional cases against our proposed framework (see Section 6).Table 2 summarizes key findings.

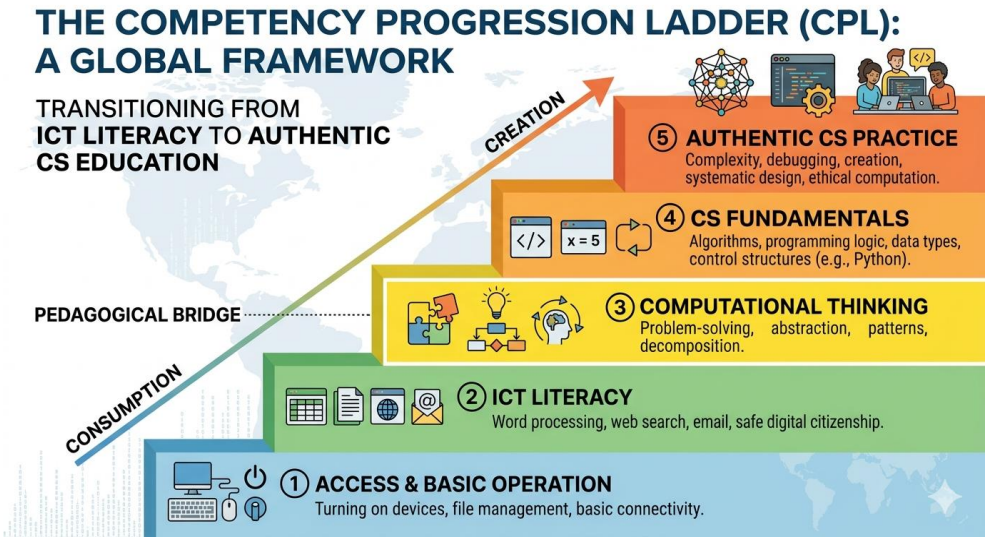
Key insight from synthesis: No single intervention works universally [37]. The most successful transitions (Estonia, Finland) combine: (a) national curriculum mandate for CS (not just ICT), (b) continuous teacher professional development (not one-off workshops), (c) locally relevant projects

(e.g., coding to solve local water or agricultural problems), and (d) bridging tools that connect block-based to text-based environments [38].

Table 2 Comparative Analysis of Six Global Interventions

| Country | Starting CPL Level | Key Intervention | Outcome | Limitation |
|---|---|---|---|---|
| Estonia | Level 2 | ProgeTiiger program: CS mandatory from Grade 1; national teacher retraining [31] | Level 5 achieved in majority of schools | Requires sustained funding for PD |
| Rwanda | Level 1 | OLPC → transition to coding clubs (Rwanda Coding Academy) [32] | Level 3 widespread; Level 4 emerging | Severe shortage of CS-trained teachers |
| India | Level 2 | CBSE CS curriculum + NGO programs (Code.org, Akshara) [33] | Elite schools: Level 5; rural schools: Level 2 | Massive within-country inequality |
| Brazil | Level 1-2 | Programaê (unplugged CT) + Brasil Mais TI initiative [34] | Strong Level 3; weak Level 4 transition | Lack of text-based coding scaffolds |
| USA | Level 2 | Exploring Computer Science + CS for All [35] | Increased equity in enrollment | Persistent race/income gap in advanced CS |
| Finland | Level 3 | CS integrated across math, crafts, and other subjects [36] | Level 5 in many schools | Teacher-reported need for more CS-specific PD |

## 6. A Proposed Global Framework: The Competency Progression Ladder (CPL)




Based on the theoretical synthesis and empirical evidence reviewed, we propose the Competency Progression Ladder (CPL) as a diagnostic and prescriptive tool for policymakers, curriculum designers, and educators.

**Level 1: Access and Basic Operation**

Competencies: Turning on device, file management, connectivity, basic navigation.

Prerequisite for: Level 2.

**Level 2: ICT Literacy**
Competencies: Word processing, spreadsheets, safe web search, email, basic digital citizenship.
Prerequisite for: Level 3.
**Level 3: Computational Thinking**
Competencies: Unplugged activities, block-based logic, sequencing, loops, conditionals (e.g., Scratch, Code.org).
Prerequisite for: Level 4.
Note: Can be taught without computers [15].
**Level 4: CS Fundamentals**
Competencies: Text-based programming: variables, data types, functions, simple algorithms (e.g., Python, JavaScript).
Prerequisite for: Level 5.
Critical transition point: Most students fail here without scaffolding [39].
**Level 5: Authentic CS Practice**
Competencies: Data structures, object-oriented design, version control, debugging complex systems, project-based creation, computational artifact evaluation.
**Transition Rules:**

- A system cannot skip levels (e.g., cannot reach Level 4 without Level 3) [40].
- Level 3 (CT) can be taught without computers, but Level 4 requires reliable hardware.
- The most common failure point is moving from Level 3 (block-based) to Level 4 (text-based). Successful transitions require scaffolded bridging tools (e.g., Pencil Code, MakeCode, dual-text/block environments) and socially relevant projects [41].

## 7. Recommendations for Policy and Practice

Drawing from the CPL framework and evidence reviewed, we offer five actionable recommendations.
### 7.1 Redefine National Digital Curricula
Explicitly replace "ICT literacy" standards with a laddered CS framework from primary through secondary education [42]. Remove conflation of word processing skills with computer science.
### 7.2 Invest in CS-Specific Teacher Preparation
Establish national CS teacher certification pathways, online micro-credentials (e.g., via CS50, Google's CS First), and long-term professional learning communities (PLCs) [43]. One-off workshops are ineffective [23].
### 7.3 Adopt "Bridging Pedagogies"
Use dual-mode programming environments (blocks-to-text) and pair programming to reduce cognitive load at the Level 3→4 transition [41]. Integrate unplugged CT activities as a prerequisite, not a substitute, for coding.
### 7.4 Design Locally Relevant Projects
Move beyond Eurocentric examples (e.g., "build a calculator") to community-grounded problems (e.g., "model crop rotation," "digitize local language folklore," "track water usage") [30, 44].
### 7.5 Tackle Affective Barriers
Explicitly counter gender stereotypes via role models, all-girls coding clubs, and assessment that rewards creative problem-solving over speed [27, 28]. Implement growth mindset interventions specific to CS [29].

## 8. Limitations and Future Research Directions

This review has several limitations. First, the predominance of English-language and OECD-country studies introduces potential publication bias [45]. Second, most included studies are cross-sectional; longitudinal evidence on transitions through all five CPL levels is scarce [46]. Third, few studies examine AI-assisted learning (e.g., ChatGPT, Copilot) as either a bridge or a crutch in CS education [47].

Future research should:

- Conduct longitudinal studies in LMICs tracking cohorts from Level 2 to Level 5 over 5-10 years.
- Develop validated, culturally-neutral assessment instruments for each CPL level.
- Investigate the role of generative AI in either accelerating or bypassing fundamental CS skill acquisition.
- Examine low-cost, offline-first solutions for Level 4 instruction in infrastructure-poor settings.

## 9. Conclusion

The digital divide will not be closed by handing out tablets loaded with office software. That approach merely creates a generation of competent consumers, not creators. The transition from ICT literacy to authentic computer science education is a matter of economic equity, cognitive justice, and democratic participation. By adopting the Competency Progression Ladder and learning from diverse global interventions, nations can build an authentic, inclusive CS education pipeline, one that empowers every student not just to use technology, but to build, critique, and reinvent it.

## References

[1] van Dijk, J. (2020). The digital divide. Polity Press.

[2] Hargittai, E. (2002). Second-level digital divide: Differences in people's online skills. First Monday, 7(4).

[3] Wei, K. K., Teo, H. H., Chan, H. C., & Tan, B. C. (2011). Conceptualizing and testing a social cognitive model of the digital divide. Information Systems Research, 22(1), 170–187.

[4] Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38–43.

[5] Warschauer, M. (2011). Learning in the cloud: How (and why) to transform schools with digital media. Teachers College Press.

[6] Code.org. (2023). 2023 State of computer science education. Code.org Advocacy Coalition.

[7] Toyama, K. (2015). Geek heresy: Rescuing social change from the cult of technology. PublicAffairs.

[8] Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering (EBSE Technical Report EBSE-2007-01). Keele University.

[9] Krumsvik, R. J. (2008). Situated learning and teachers' digital competence. Education and Information Technologies, 13(4), 279–290.

[10] Claro, M., Preiss, D. D., San Martín, E., Jara, I., Hinostroza, J. E., Valenzuela, S., ... & Nussbaum, M. (2012). Assessment of 21st century ICT skills in Chile. Computers & Education, 59(3), 1042–1053.

[11] Eshet-Alkalai, Y. (2004). Digital literacy: A conceptual framework for survival skills in the digital era. Journal of Educational Multimedia and Hypermedia, 13(1), 93–106.

[12] International ICT Literacy Panel. (2002). Digital transformation: A framework for ICT literacy. Educational Testing Service.

[13] Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.

[14] Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books.

[15] Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Proceedings of the 2012 Annual Meeting of the American Educational Research Association, 1–25.

[16] Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. Proceedings of the 16th Koli Calling International Conference on Computing Education Research, 120–129.

[17] Hilbert, M. (2011). The end justifies the definition: The manifold outlooks on the digital divide and their practical usefulness for policy-making. Telecommunications Policy, 35(8), 715–736.

[18] James, J. (2012). The digital divide beyond the access divide. Information Technology for Development, 18(4), 279–282.

[19] UNESCO. (2021). Global education monitoring report 2021: Technology in education. UNESCO Publishing.

[20] DiMaggio, P., & Hargittai, E. (2001). From the 'digital divide' to 'digital inequality'. Princeton University Center for Arts and Cultural Policy Studies Working Paper, 15, 1–23.

[21] Rafalow, M. H. (2018). Disciplining play: Digital youth culture as capital at school. American Journal of Sociology, 123(5), 1416–1452.

[22] Ni, L., & Guzdial, M. (2019). Who am I? Understanding high school computer science teachers' professional identity. ACM Transactions on Computing Education, 19(4), 1–23.

[23] Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: Understanding teacher experiences and challenges. Computer Science Education, 26(4), 235–254.

[24] ACM (Association for Computing Machinery). (2022). 2022 ACM K–12 CS teacher survey report. ACM.

[25] Okonkwo, C. W., & Ade-Ibijola, A. (2021). Python programming in Nigerian secondary schools: Teachers' perspectives. Education and Information Technologies, 26(4), 4325–4348.

[26] Master, A., Cheryan, S., & Meltzoff, A. N. (2016). Computing whether she belongs: Stereotypes undermine girls' interest and sense of belonging in computer science. Journal of Educational Psychology, 108(3), 424–437.

[27] Cheryan, S., Ziegler, S. A., Montoya, A. K., & Jiang, L. (2017). Why are some STEM fields more gender balanced than others? Psychological Bulletin, 143(1), 1–35.

[28] Google & Gallup. (2019). Diversity gaps in computer science education. Google Inc.

[29] Yeager, D. S., & Dweck, C. S. (2012). Mindsets that promote resilience. Educational Psychologist, 47(4), 302–314.

[30] Walton, M. (2019). Decolonizing computer science education. Journal of Learning for Development, 6(3), 220–236.

[31] Laanpere, M., & Põldoja, H. (2018). The ProgeTiiger programme in Estonia. In A. Tatnall (Ed.), Encyclopedia of education and information technologies (pp. 1–6). Springer.

[32] Nizeyimana, G., & Nsengimana, T. (2020). Coding clubs in Rwanda: From OLPC to digital skills. African Journal of ICT, 12(1), 45–62.

[33] Sharma, K., & Singh, R. (2019). Computer science education in India: A tale of two schools. Indian Journal of Educational Technology, 4(2), 88–104.

[34] Brackmann, C. P., Roman-Gonzalez, M., Robles, G., Moreno-Leon, J., Casali, A., & Barone, D. (2017). Development of computational thinking skills through unplugged activities in primary school. Proceedings of the 12th Workshop on Primary and Secondary Computing Education, 65–72.

[35] Margolis, J., Goode, J., & Chapman, G. (2017). Stuck in the shallow end: Education, race, and computing. MIT Press.

[36] Lintilä, T., & Vesisenaho, M. (2021). Integrating computer science into Finnish basic

education. Nordic Journal of Digital Literacy, 16(2), 58–72.

[37] Unwin, T. (2017). Reclaiming information and communication technologies for development. Oxford University Press.

[38] Weintrop, D., & Wilensky, U. (2019). Transitioning from block-based to text-based programming languages. Journal of Computer Science Integration, 2(1), 1–28.

[39] Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. Communications of the ACM, 60(6), 72–80.

[40] Resnick, M., & Robinson, K. (2017). Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play. MIT Press.

[41] Franklin, D., Hill, C., Dwyer, H., Hansen, A., & Iveland, A. (2020). Bridging the block-to-text gap. ACM Transactions on Computing Education, 20(4), 1–25.

[42] European Commission. (2019). *Digital education action plan 2021-2027*. Publications Office of the European Union.

[43] Mouza, C., Yang, H., Pan, Y. C., Ozden, S. Y., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers. Journal of Technology and Teacher Education, 25(1), 81–110.

[44] Eglash, R., Bennett, A., O'Donnell, C., Jennings, S., & Cintorino, M. (2006). Culturally situated design tools: Ethnocomputing from field site to classroom. American Anthropologist, 108(2), 347–362.

[45] Okoli, C., & Schabram, K. (2010). A guide to conducting a systematic literature review of information systems research. Sprouts: Working Papers on Information Systems, 10(26), 1–49.

[46] Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking. Computers & Education, 78, 348–364.

[47] Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Programming is hard, or at least it used to be. Proceedings of the 54th ACM Technical Symposium on Computer Science Education, 500–506.