Designing Transformative Learning Environments: Embedding Computational Thinking into Digital Education

Oliver Kastner-Hauler[1][0000−0002−9958−3298], Bernhard Standl[2][0000−0002−8849−2980], Barbara Sabitzer[3][0000−0002−1304−6863], and Zsolt Lavicza[3][0000−0002−3701−5068]

[1] University of Education Lower Austria, Department of Media Education, 2500 Baden, Austria oliver.kastner@ph-noe.ac.at
[2] Karlsruhe University of Education, Department of Informatics and Digital Education, 76133 Karlsruhe, Germany bernhard.standl@ph-karlsruhe.de
[3] Johannes Kepler University Linz, Department of STEM Education, 4040 Linz, Austria {barbara.sabitzer,zsolt.lavicza}@jku.at

Abstract Integrating computational thinking (CT) into school curricula is increasingly recognized as essential for preparing students for the digital age. Austria's introduction of basic digital education (BDE) in 2018 and revised in 2022 highlighted this need. However, teachers without a computer science (CS) background often struggle to teach CT, focusing instead on media and computer literacy, which underemphasizes CT in BDE. To address this issue, we developed design principles for a handbook through three practical learning environments (LEs) using physical computing with single-board computing devices. This approach, grounded in constructionist theory, emphasizes experiential learning to foster intrinsic understanding of CS/CT concepts. Additionally, an Open Educational Resource (OER) textbook was redesigned to align with the 2022 curriculum and the 5E instructional model, promoting self-directed, inquiry-based learning. The 2022 curriculum is based on the 'Frankfurt-Dreieck' model, representing three facets of digitization. To operationalize these multi-faceted requirements, we synthesized these theoretical perspectives into a cohesive multi-resource approach. This strategy aims to make CS/CT concepts more accessible to teachers and students, regardless of prior CS knowledge, translating CT for the classroom. Through the implementation of these design principles, educators are systematically guided to gain confidence in teaching CT, ensuring its thorough integration into BDE. Future research will focus on evaluating and refining these principles on a larger scale to develop a comprehensive handbook. Comparative studies with neighboring European countries are planned to identify best practices and enhance the effectiveness of CT integration in K-8 education. This ongoing work aims to provide robust support for educators and foster a generation of students well-prepared for the demands of the digital age.

Keywords: Computational Thinking · Digital Education Curriculum · Design Principles · Physical Computing · Practical Handbook.

1 Introduction

Computational thinking (CT) has become a vital component in preparing students with the skills necessary for the evolving demands of future careers [15,52]. CT entails the application of methods and strategies from the domain of computer science (CS) to solve problems across a diverse range of disciplines, extending beyond the traditional scope of CS [54]. In Austria, the overarching concept of Basic Digital Education (BDE) integrates CT with media literacy and computer proficiency. Since its inception as a compulsory exercise in 2018, BDE has undergone significant development, culminating in its establishment as a mandatory subject for students aged 10-14 in grades 5-8 in 2022. This transformation included a complete curricular overhaul based on the 'Frankfurt-Dreieck' model [5]. The 'Frankfurt-Dreieck' (Frankfurt Triangle) model has gained prominence in educational curriculum design, offering a structured framework for understanding the interplay of technology, society, and application in the digital age [8]. This model aligns well with CT and BDE objectives, promoting a multifaceted understanding of digital systems and their impacts. However, viewing BDE through this lens introduces new challenges for educators. Despite professional development opportunities, teachers who feel underprepared may focus on familiar BDE aspects, potentially neglecting CT and the fundamental components of every digital education. Our research aims to establish guiding design principles for content curation and elaboration for K-8 students aged 10-14, with the goal of developing a solid practice-oriented handbook. This resource will equip educators with the tools to plan and deliver, and students to engage with, a comprehensive digital education that integrates all critical facets of the 'Frankfurt-Dreieck' model. Our endeavor is grounded in a thorough analysis of existing literature and curricula, employing a design-based research (DBR) approach [32,33]. Three learning environments (LEs) have been developed that facilitate effective teaching and learning of CT. These LEs support educators and learners alike in their initial exploration of the subject, even without prior expertise in CS [6,24,25]. The design principles derived from these LEs empower instructors to create new or enhance existing BDE materials, regardless of their formal CS background [20]. By following the principles, teachers will be assured that they are fulfilling the curriculum aligned with the 'Frankfurt-Dreieck' without undue concerns about the finer details. By removing the requirement of prior CS knowledge and incorporating an engaging, playful approach, we have identified a promising strategy to lower the barriers for both students and teachers to engage with CT in the classroom. This approach serves as a crucial link, bridging the gap between media literacy, computer proficiency, and informatics, which includes both CS and CT. Such a methodology is essential to ensure the effective implementation of comprehensive digital education in modern schools [12].

2 Background

The conceptualization of the elements that comprise computational thinking (CT) is an ongoing process of refinement and evolution [48]. Palts and Pedaste [39] present a graphical overview of the evolution of CT definitions over time, which provides a useful illustration of the situation. Analogous portrayals of CT assessments within the field were researched by Tang et al. [49]. Li et al. [31] perceive CT as a cognitive framework for reasoning and problem-solving, transcending mere computing proficiency. Embracing this perspective, the guiding principles for our handbook have been designed to accommodate future advancements in CT. The dynamic nature of computing can facilitate interdisciplinary adoption of CT across STEAM disciplines [43].

The synthesis of CT competencies with block-based programming, physical computing, and inquiry-based learning, situated within the constructivist pedagogical framework [41], serves as the cornerstone of the principles that guide our handbook. Grounded in this pedagogical framework, three learning environments were developed, applying the 3D Framework of 4Ps [7] merged with COOL Informatics [47]. The LEs were iteratively refined to match the didactic approaches and competence areas of the BDE curriculum and to reflect the multi-perspective view of the 'Frankfurt-Dreieck' on digital education. The following sections provide a more comprehensive examination of the fundamental elements that form the basis of this research.

2.1 Synthesizing Block-based Programming, Physical Computing, and Inquiry-Based Learning

Block-based programming languages, commonly used in lower grades due to their low entry barriers, offer a hands-on approach to learning by enabling students to visually create code through block objects that contain instructions, much like building a toy house with snap-together construction bricks [53]. Even in early childhood, this approach is particularly beneficial for the development of problem-solving and associated cognitive skills [40]. In the LEs discussed in this paper, the official block-based environment Makecode for micro:bit [34,35] was employed to solve problems and code a solution. The choice of this platform supports Papert's view of coding or programming as a vital element to intellectual development [27] and is complemented by the concept of physical computing. Physical computing is the concept of connecting devices to the physical environment, fostering a deeper understanding of programming and encouraging learners to manifest their problem-solving thought processes through tangible products [42]. This haptic interaction has been shown to expand learners' creativity and imagination about the possibilities of code [45]. To facilitate these learning experiences, we employ inquiry-based learning (IBL) that utilizes the 5E instructional model –*engage, explore, explain, elaborate, and evaluate*– as a flexible framework for fostering curiosity and deeper investigation [1,44]. This model is used to structure the process, with learners engaging interactively with

resources designed to encourage playful exploration and active inquiry, either through guided or self-directed discovery [18].

2.2 Curriculum: Didactic Approaches and Competence Areas

The implementation of basic digital education (BDE) necessitates holistic **didactic approaches** to digital artifacts described in the following by the curriculum [5]. These approaches encompass co-constructive, experiential, design-oriented, reflective, and problem-solving methods. Critical thinking, design thinking, inquiry-based learning, and playful learning are highlighted as key methodologies. BDE demands interdisciplinary and transdisciplinary forms of classroom work, integrating the three primary domains:

Media Education focuses on the genesis, evolution, and future of digital media constellations. It involves reflection on media-biographical developments, conditions of media socialization, and digital inclusion/exclusion dynamics. **Informatics Education** encompasses analyzing, interacting, modeling, coding, and testing in relation to informatics systems, software, automation, data, and networking. It aligns with 21st-century skills, emphasizing critical thinking, creativity, communication, collaboration, and computational thinking. **Design Competence** integrates informatics education and media education, offering diverse analytical, productive, and creative approaches to functional media applications and aesthetic media formats in globalized digital cultures.

These three domains are intended to be interwoven in a balanced, creative, and integrative manner to foster a comprehensive digital literacy [23].

The **competence areas** to apply from the curriculum [5] are divided into five main sections, each with specific goals for each grade level in middle school from grade 5-8 (10-14 years):

Orientation: Analyzing and reflecting on the social aspects of media change and digitization. **Information:** Handling data, information, and information systems responsibly. **Communication:** Using informational and media systems to communicate and cooperate effectively. **Production:** Creating and publishing digital content, designing algorithms, and programming. Identify, develop, and apply computational thinking skills to solve problems. **Action:** Evaluating and responsibly utilizing the offers and options available in a digital world.

2.3 'Frankfurt-Dreieck' Model

The 'Frankfurt-Dreieck' model, proposed by Brinda et al. [8], offers a comprehensive framework for understanding and implementing education in a digital world. The interdisciplinary approach of the model emphasizes the interplay between three key perspectives: the technological-medial (T), socio-cultural (G)[4], and interaction-oriented (I) dimensions of digital phenomena and is part of the

[4] Society (German): (G)esellschaft

curriculum for BDE [5]. The model's strength lies in its ability to contextualize technological concepts within societal impacts and practical utilization, thereby providing educators with a more nuanced and interdisciplinary understanding of digital systems. This approach is especially beneficial in the context of the accelerated digital transformation that is currently underway. It equips students with the analytical skills to not only create and utilize technology but also to critically evaluate its implications and potential consequences in a variety of contexts.

The model comprises three central perspectives on the digitization of education – see Figure 1: **(T) Technological-medial** analyzes the role of technologies and media in education and communication. **(G)**[1] **Socio-cultural** looks at the impact of digital transformation on society and culture, including social inequalities and changes in values and norms. **(I) Interaction-oriented** analyzes and reflects upon the interaction between individuals and digital systems during application. The 'Frankfurt-Dreieck' [8] is aimed at researchers who deal with the theory and reflection of education in the digital context. It is intended to contribute to the development of a comprehensive and multi-perspective view on the challenges and opportunities of digital transformation for education. It was developed by computer scientists, media educators, and media researchers and is based on the "Dagstuhl-Dreieck – Declaration on Education in the Digital Age"[36].

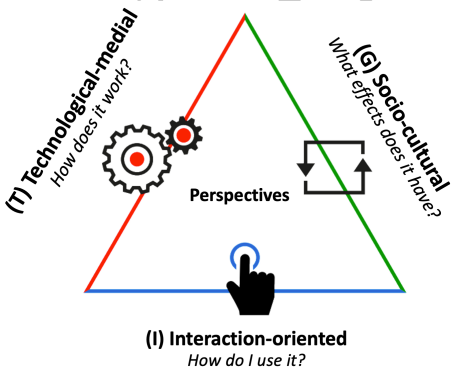

Figure 1. 'Frankfurt-Dreieck' adaptation from Doebeli Honegger & Salzmann [13] with extensions by the first author.

3 Methods

This study attempts to achieve a milestone in a comprehensive design-based research (DBR) project [33] aimed at developing practical guidelines for a hands-on teaching and learning handbook for CT. Launched in 2019, the project introduced the micro:bit and computational thinking (CT) concepts to primary and

secondary schools, focusing on teachers and students aged 8-14. The milestone encompasses three key investigations: the adaptation of an OER for flipped learning [24], an exploration of the combined effects of physical computing and block-based programming in primary schools [25], and an assessment of CT teaching effectiveness in middle schools using Bebras [26]. Participating teachers received specialized training to implement these learning environments (LEs) and the corresponding assessment tools effectively.

To achieve a better balance between digital media, computer literacy, and informatics (CS/CT) for the new BDE curriculum, we investigated programming within the broader context of media use and creation. Resnick and Rusk [46] identified 4Ps – the four pillars of effective digital education –*projects, passion, peers, and play*– emphasizing meaningful projects, student engagement, collaboration, and playful experimentation. Similar principles are found in the COOL Informatics approach [47], which promotes discovery, cooperation, individuality, and activity. COOL Informatics also integrates neurodidactical perspectives, aligning closely with the 4Ps framework [23].

By transforming these approaches, we developed three CT-focused LEs that form the foundation of practical handbook principles, offering actionable strategies for the classroom. Our analysis highlighted a strong synergy between the 4Ps framework and COOL Informatics, particularly in the alignment of 'peers' from the 4Ps with 'cooperation' in COOL Informatics. The principles derived from this research encompass all elements of the 4Ps, creating a clear progression from theory to practice. The resulting framework serves as a guide to enhance understanding and awareness of CT in the classroom, establishing a comprehensive link between computational thinking, media literacy, and computer skills for basic digital education (BDE). Building on a previous conference paper [23], we expanded our framework to further explore curriculum alignment, focusing on mapping the LEs to the curriculum's two-dimensional competency framework. As demonstrated in Appendix - Table 1, our framework provides thorough and balanced coverage of nearly all facets of the curriculum.

4 Results

This section outlines the derived design principles for creating effective learning environments (LEs) for CT and problem-solving to scaffold teachers, especially those new to computer science, as they embed CT into BDE lessons. The following eight principles facilitate the 4Ps and COOL Informatics frameworks, allowing for smooth classroom implementation and curriculum-aligned lesson planning.

'Hello World': Asking *why* and *what for* fosters understanding of programming concepts, but quick action is crucial in the beginning. Typically, beginners start with the basic task to output 'Hello World' on the display. With Makecode and micro:bit, learners can easily start by making a heart appear on the 5x5 LED display instead of text. The color-coded programming blocks are easy

to find, and learners can playfully create programs by dragging and dropping blocks, experimenting until they reach the desired result.

Input-Process-Output (IPO): The IPO principle demonstrates how computers operate. We introduce input using micro:bit's A and B buttons after showing output with a heart on the display. Alternately pressing A (display heart) and B (clear) simulates a flashing heart. Moving on, we use traffic lights as an analogy for automated processing. To automate switching and flashing, we implement button processing via a loop. Pressing A+B initiates the process, alternating between heart and blank displays. This creates an automated *flashing heart*, preparing the next steps for evaluation and testing.

Evaluation and Debugging: Initial testing may reveal that the blinking occurs only once when triggered by A+B, highlighting the importance of evaluation in programming. When outcomes do not meet intentions, debugging is crucial to identify and fix code flaws [28]. Makecode's step-by-step debugger and program flow verbalization can aid this process [19]. Pair programming principles should be applied to evaluation, debugging, and the entire computational thought process.

Pair Programming: Budget constraints often prevent providing one device per child in new projects. Pair programming offers both a solution and educational benefits by having two programmers collaborate on one device. One types (driver), while the other guides (navigator) [9]. This method fosters communication and understanding, with regular role reversal enhancing code quality and peer interaction [17]. Integrating peer learning and teaching with open-ended tasks from the textbook wiki [29] and using animated tutorials from Makecode can further enrich the experience by demonstrating code from a third-person perspective.

Open-Ended Learning and Makerspaces: The OER textbook encourages open-ended learning, fostering creativity, collaboration, and critical thinking in CT [51]. It recommends sample extensions for further micro:bit project development. Practical experiences in makerspaces, games, or real-world challenges deepen knowledge. Makerspace activities fit well into flipped classrooms, when split into pre-class programming and in-class making to maximize hands-on time [24]. This approach fosters a growth mindset, ownership of learning, and new opportunities through challenges in physical computing.

Physical Computing and AHA! Experience: Connecting a micro:bit to a computer offers a tactile experience of code and algorithms [22]. Online tutorials from the textbook wiki or Makecode provide initial guidance. Once code upload is mastered, a portable power supply may be needed, marking the first "AHA!" moment as concepts become tangible [50]. With onboard sensors and actuators, the micro:bit interacts with the physical world, such as a portable step counter. For further exploration, attach it to a leg to count steps using the accelerometer and a portable power supply [38].

CT-Dimensions: Concepts, Practices, and Perspectives: Educators need a deep understanding of Brennan and Resnick's 3D framework [7] to effectively design and assess CT learning experiences. The Beginners Computational

Thinking test (BCTt) can assess foundational *CT concepts* (sequences, loops, conditionals, data, operators) essential for coding [56]. While CT assessments were explored in prior studies [6,25], this paper focuses on handbook principles. CT practices and perspectives cannot be directly measured from Makecode artifacts. We analyzed the LEs for *CT practices* (like evaluation and debugging) and *CT perspectives* through feedback on pair programming and CS-unplugged activities. By evaluating the LEs comprehensively, we indirectly assess CT practices and perspectives.

CS-Unplugged Activities: Use CS-Unplugged activities to teach CS basics without computers [4]. This free resource uses games and everyday items to explain CS concepts. For example, the parity magic trick can teach error detection and correction, a crucial data transmission concept. CS-Unplugged offers a unique, hands-on approach to learning foundational CS and CT principles, essential for digital education [10].

5 Discussion and Conclusion

Computational thinking (CT) within the BDE curriculum corresponds to the technological-medial (T) aspect of the 'Frankfurt-Dreieck' described in Section 2.3. Furthermore, the curriculum contains five competence areas (orientation, information, communication, production, action) as the second dimension – see Section 2.2. However, the curriculum's lack of explicitness regarding actionable CT can hinder teachers' readiness to incorporate CT into their lessons. To address this, our approach advocates for a smooth introduction to computer science (CS) and CT through the implementation of our design principles for a practical handbook, based on educational design research (EDR) [3,32]. This will make CT more accessible for new BDE teachers, strengthening their confidence and facilitating learning of basic CS/CT concepts through hands-on experience. By focusing on the principles, teachers can better understand how CT relates to their broader educational goals of BDE and implement it effectively in their classrooms.

The emerged design principles for CT education emphasize core programming concepts, including 'Hello World,' Input-Process-Output, and debugging. These foundational practices foster a problem-solving mindset, requiring reflective thinking and initial guidance, complemented by scaffolding material [51]. Integrating open-ended challenges alongside makerspaces and physical computing demands precise implementation guidelines [14]. The holistic view of CT from the 3D framework unifies all dimensions of a comprehensive CT education and enhances its accessibility. This overarching concept permeates all materials and can be highlighted through CS-unplugged activities. A practical implementation example of the design principles is included in the Appendix.

Based on Hsu's [21] review, our handbook design principles for CT draw on effective learning and teaching strategies and best practices. This provides a strong foundation for educators aiming to integrate CT into their classrooms, regardless of a formal CS background.

Future research will focus on evaluating and refining these principles through in-service teacher training on basic digital education (BDE) to assess their impact on the acceptance of CT within BDE and for future classroom integration. By examining how teachers adapt and create learning materials based on these principles, we can develop a comprehensive support handbook, refining its content and design through iterative reflections. Moreover, by conducting comparative studies with neighboring European countries [30], we aim to identify best practices in CT integration for K-8 education in Austria and abroad.

Conclusion. Integrating computational thinking (CT) into classrooms provides a valuable opportunity for its continued development through experiential learning. By transforming LEs to seamlessly embed CT constructs like algorithms, abstraction, and automation, students and teachers can deepen their understanding of fundamental computer science (CS) principles and cultivate CT skills without pre-existing knowledge [55]. As students and teachers engage with CT and apply its problem-solving approaches, they naturally develop confidence and fluency to master the complex challenges of the digital era.

Acknowledgements

Data Availability. The article includes the original contributions presented in the study, further inquiries can be directed to the corresponding author.

Ethics. The research involving human participants was reviewed and approved by all the universities of the authors. Written informed consent to participate in this study was obtained from the participants' legal guardian/next of kin.

Disclosure statement. No potential conflict of interest was reported by the authors.

Funding. No funding was used for this research.

Artificial Intelligence (AI). AI-tools assisted in language refinement and clarity enhancement under human supervision. DeepL [11] improved translations and style, while ChatGPT [37], Claude [2], and Gemini [16] aided in text refinement and shortening. All original ideas, analysis, interpretations, and conclusions remain the sole responsibility of the authors.

References

1. Ah-Nam, L., Osman, K.: Developing 21st century skills through a constructivist-constructionist learning environment. K-12 Stem Education **3**(2), 205–216 (2017). https://doi.org/10.14456/K12STEMED.2017.6
2. Anthropic: Claude, https://www.anthropic.com, Large Language Model
3. Bakker, A.: Design research in education: A practical guide for early career researchers. Routledge (2018)
4. Bell, T., Vahrenhold, J.: CS Unplugged - How Is It Used, and Does It Work? In: Adventures Between Lower Bounds and Higher Altitudes. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98355-4_29

5. BMBWF: Lehrplan Digitale Grundbildung. Bundesministerium für Bildung Wissenschaft und Forschung (2022), https://www.ris.bka.gv.at/Dokumente/BgblAuth/BGBLA_2022_II_267/BGBLA_2022_II_267.pdfsig
6. Brandhofer, G., Kastner-Hauler, O.: Der micro:bit und Computational Thinking. Evaluierungsergebnisse zu einem informatischen Projekt. R&E-SOURCE (2020), https://journal.ph-noe.ac.at/index.php/resource/article/view/914
7. Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking. In: 2012 AERA Proceedings. vol. 1, p. 25. Vancouver, Canada (2012), https://scratched.gse.harvard.edu/ct/files/AERA2012.pdf
8. Brinda, T., Brüggen, N., Diethelm, I., Knaus, T., Kommer, S., Kopf, C., Missomelius, P., Leschke, R., Tilemann, F., Weich, A.: Frankfurt-Dreieck zur Bildung in der digital vernetzten Welt. GI (2019). https://doi.org/10.25656/01:22117
9. Bryant, S., Romero, P., Du Boulay, B.: Pair programming and the mysterious role of the navigator. International Journal of Human-Computer Studies **66**(7), 519–529 (2008). https://doi.org/10.1016/j.ijhcs.2007.03.005
10. Curzon, P., McOwan, P.W.: Computational Thinking: Die Welt des algorithmischen Denkens – in Spielen, Zaubertricks und Rätseln. Springer, Berlin, Heidelberg (2018). https://doi.org/10.1007/978-3-662-56774-6
11. DeepL GmbH: Translator & Write, https://www.deepl.com, online machine translation service, AI-powered writing assistant
12. Diethelm, I.: Digital Education and Informatics – You can't have One without the Other. In: Proceedings of the 17th Workshop in Primary and Secondary Computing Education. pp. 1–2. ACM (2022). https://doi.org/10.1145/3556787.3556790
13. Doebeli Honegger, B., Salzmann, R.: Daggstuhl-Dreieck (2018), https://mia.phsz.ch/Dagstuhl/GrafikUnterCCLizenz, CC BY-SA 4.0
14. Emara, M., Hutchins, N., Grover, S., Snyder, C., Biswas, G.: Examining Student Regulation of Collaborative, Computational, Problem-Solving Processes in Open-Ended Learning Environments. Journal of Learning Analytics **8**(1), 49–74 (2021). https://doi.org/10.18608/jla.2021.7230
15. Fraillon, J., Ainley, J., Schulz, W., Friedman, T., Duckworth, D.: Preparing for Life in a Digital World: IEA International Computer and Information Literacy Study 2018 International Report. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38781-5
16. Google: Gemini, https://gemini.google.com, Large Language Model
17. Graßl, I., Fraser, G.: The ABC of Pair Programming: Gender-dependent Attitude, Behavior and Code of Young Learners. In: ICSE 2023 Proceedings. p. 13 (2023). https://doi.org/10.48550/arXiv.2304.08940
18. Guzdial, M.: Learner-Centered Design of Computing Education: Research on Computing for Everyone. Synthesis Lectures on Human-Centered Informatics, Springer, Cham (2015). https://doi.org/10.1007/978-3-031-02216-6
19. Heikkilä, M., Mannila, L.: Debugging in Programming as a Multimodal Practice in Early Childhood Education Settings. Multimodal Technologies and Interaction **2**(3), 42 (2018). https://doi.org/10.3390/mti2030042
20. Hromkovič, J., Lacher, R.: How to convince teachers to teach computer science even if informatics was never a part of their own studies. Bulletin of the EATCS **123**, 6 (2017), https://api.semanticscholar.org/CorpusID:4795189
21. Hsu, T.C., Chang, S.C., Hung, Y.T.: How to learn and how to teach computational thinking: Suggestions based on a review of the literature. Computers & Education **126**, 296–310 (2018). https://doi.org/10.1016/j.compedu.2018.07.004

22. Kalelioglu, F., Sentance, S.: Teaching with physical computing in school: the case of the micro:bit. Education and Information Technologies **25**(4), 2577–2603 (2020). https://doi.org/10.1007/s10639-019-10080-8
23. Kastner-Hauler, O., Standl, B., Sabitzer, B., Lavicza, Z.: Developing design principles for computational thinking learning environments: Pathways into practice with physical computing:. In: 2024 Proceedings of CSEDU Conference. pp. 445–453. SCITEPRESS (2024). https://doi.org/10.5220/0012689600003693
24. Kastner-Hauler, O., Tengler, K., Demarle-Meusel, H., Sabitzer, B.: Adapting an OER textbook for the inverted classroom model - how to flip the classroom with BBC micro:bit example tasks. In: 2021 FIE Conference. pp. 1–8. IEEE (2021). https://doi.org/10.1109/FIE49875.2021.9637170
25. Kastner-Hauler, O., Tengler, K., Sabitzer, B., Lavicza, Z.: Combined effects of block-based programming and physical computing on primary students' computational thinking skills. Frontiers in Psychology **13**, 875382 (2022). https://doi.org/10.3389/fpsyg.2022.875382
26. Kastner-Hauler, O., Tengler, K., Sabitzer, B., Lavicza, Z.: A learning environment to promote the computational thinker: A bebras perspective evaluation. In: Pluhár, Z., Gaál, B. (eds.) Informatics in Schools. Innovative Approaches to Computer Science Teaching and Learning, ISSEP 2024, vol. 15228, pp. 85–98. Springer Nature Switzerland (2025). https://doi.org/10.1007/978-3-031-73474-8_7
27. Kestenbaum, D.: The challenges of IDC: what have we learned from our past? Communications of the ACM **48**(1), 35–38 (2005). https://doi.org/10.1145/1039539.1039566
28. Kim, C., Yuan, J., Vasconcelos, L., Shin, M., Hill, R.B.: Debugging during block-based programming. Instructional Science **46**(5), 767–787 (2018). https://doi.org/10.1007/s11251-018-9453-5
29. Kröhn, C., Sabitzer, B.: Peer-learning and Talents Exchange in Programming: Experiences and Challenges:. In: 2020 Proceedings CSEDU. pp. 466–471. SCITEPRESS (2020). https://doi.org/10.5220/0009472004660471
30. Kvaššayová, N., Mansell, M., Cápay, M., Bellayová, M.: The BBC micro:bit in Slovakia. In: 2021 IDT Conference. pp. 359–365 (2021). https://doi.org/10.1109/IDT52577.2021.9497573
31. Li, Y., Schoenfeld, A.H., diSessa, A.A., Graesser, A.C., Benson, L.C., English, L.D., Duschl, R.A.: Computational Thinking Is More about Thinking than Computing. Journal for STEM Education Research **3**(1), 1–18 (2020). https://doi.org/10.1007/s41979-020-00030-2
32. McKenney, S., Reeves, T.C.: Conducting educational design research. Routledge, London, 2nd edn. (2018). https://doi.org/10.4324/9781315105642
33. McKenney, S., Reeves, T.C.: Educational design research: Portraying, conducting, and enhancing productive scholarship. Medical Education **55**, 82–92 (2021). https://doi.org/10.1111/medu.14280
34. Microbit Foundation: The Micro:bit Educational Foundation, https://microbit.org/about/
35. Microsoft: Microsoft MakeCode for micro:bit, https://makecode.microbit.org/, https://makecode.microbit.org/, [Accessed 2023-11-11]
36. Missomelius, P.: Die Dagstuhl-Erklärung: Erklärung zur relevanz von medienbildung. Medienimpulse **54**(1) (2016). https://doi.org/10.21243/mi-01-16-15
37. OpenAI: Chatgpt-4, https://openai.com/gpt-4, Large Language Model
38. O'Sullivan, D., Igoe, T.: Physical computing: sensing and controlling the physical world with computers. Thomson Publishing, Boston (2004)

39. Palts, T., Pedaste, M.: A Model for Developing Computational Thinking Skills. Informatics in Education **19**(1), 113–128 (2020). https://doi.org/10.15388/infedu.2020.06
40. Papadakis, Stamatios, P.: Can Preschoolers Learn Computational Thinking and Coding Skills with ScratchJr? A Systematic Literature Review. International Journal of Educational Reform pp. 1–34 (2022). https://doi.org/10.1177/10567879221076077
41. Papert, S.: Mindstorms; Children, Computers and Powerful Ideas. Basic Books, New York, NY (1980)
42. Papert, S., Harel, I.: Situating constructionism. Constructionism **36**(2), 1–11 (1991), http://www.papert.org/articles/SituatingConstructionism.html
43. Pears, A., Barendsen, E., Dagienė, V., Dolgopolovas, V., Jasutė, E.: Holistic STEAM Education Through Computational Thinking: A Perspective on Training Future Teachers. In: ISSEP 2019, vol. 11913, pp. 41–52. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33759-9_4
44. Pedaste, M., Mäeots, M., Siiman, L.A., de Jong, T., van Riesen, S.A., Kamp, E.T., Manoli, C.C., Zacharia, Z.C., Tsourlidaki, E.: Phases of inquiry-based learning: Definitions and the inquiry cycle. Educational Research Review **14**, 47–61 (2015). https://doi.org/10.1016/j.edurev.2015.02.003
45. Przybylla, M., Romeike, R.: Physical Computing and its Scope - Towards a Constructionist Computer Science Curriculum with Physical Computing. Informatics in Education **13**(2), 225–240 (2014). https://doi.org/10.15388/infedu.2014.14
46. Resnick, M., Rusk, N.: Coding at a crossroads. Communications of the ACM **63**(11), 120–127 (2020). https://doi.org/10.1145/3375546
47. Sabitzer, B., Demarle-Meusel, H., Painer, C.: A COOL Lab for Teacher Education. In: Rethinking Teacher Education for the 21st Century: Trends, Challenges and New Directions, pp. 319–328. Verlag Barbara Budrich, Leverkusen, Toronto (2019)
48. Selby, C., Woollard, J.: Computational Thinking: The Developing Definition (2013), https://eprints.soton.ac.uk/356481/1/Selby_Woollard_bg_soton_eprints.pdf
49. Tang, X., Yin, Y., Lin, Q., Hadad, R., Zhai, X.: Assessing computational thinking: A systematic review of empirical studies. Computers & Education **148**, 103798 (2020). https://doi.org/10.1016/j.compedu.2019.103798
50. Thagard, P., Stewart, T.C.: The AHA! Experience: Creativity Through Emergent Binding in Neural Networks. Cognitive Science **35**(1), 1–33 (2011). https://doi.org/10.1111/j.1551-6709.2010.01142.x
51. Tsan, J., Eatinger, D., Pugnali, A., Gonzalez-Maldonado, D., Franklin, D., Weintrop, D.: Scaffolding Young Learners' Open-Ended Programming Projects with Planning Sheets. In: 2022 Proceedings ITICSE Conference. pp. 372–378. ACM (2022). https://doi.org/10.1145/3502718.3524769
52. WE Forum: The Future of Jobs Report 2023. World Economic Forum (2023), https://www.weforum.org/reports/the-future-of-jobs-report-2023/
53. Weintrop, D., Wilensky, U.: Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. ACM Transactions on Computing Education **18**(1), 1–25 (2017). https://doi.org/10.1145/3089799
54. Wing, J.M.: Computational Thinking. Communications of the ACM **49**(3), 33–35 (2006). https://doi.org/10.1145/1118178.1118215
55. Yadav, A., Hong, H., Stephenson, C.: Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. TechTrends **60**, 565–568 (2016). https://doi.org/10.1007/s11528-016-0087-7

56. Zapata-Cáceres, M., Martín-Barroso, E., Román-González, M.: BCTt: Beginners Computational Thinking Test. In: Understanding computing education, pp. 46–56. Raspberry Pi Foundation (2021), https://www.raspberrypi.org/app/uploads/2021/05/Understanding-computing-education-Volume-1-%E2%80%93-Raspberry-Pi-Foundation-Research-Seminars.pdf

APPENDIX

Implementation of CT Handbook Principles – Example

Mr. John Mayer, a middle school teacher for grade 8 students, organizes a project to introduce computational thinking (CT) through environmental data collection and analysis using the micro:bit and external sensors. The project involves logging temperature and humidity data from indoor and outdoor locations, transmitting this data via radio to a central logger.

'Hello World': Students begin by programming a micro:bit to display temperature readings on its LED screen. This simple task familiarizes them with the Makecode environment and provides an early success. **Input-Process-Output:** Mr. Mayer explains IPO using the micro:bit's built-in temperature sensor as input, data processing in the code, and LED display or radio transmission as output. **Debugging and Pair Programming:** Students work in pairs to develop their data logging programs. One student writes code while the other reviews and debugs, enhancing their problem-solving skills and code quality. **Open-ended Learning and Inquiry-Based Learning:** Mr. Mayer challenges students to design an efficient data logging system. Students research micro:bit capabilities, radio communication protocols, and data storage methods, fostering self-directed learning. **Physical Computing and AHA! Experience:** Students experience tactile "AHA!" moments when they connect external sensors to the micro:bit and successfully transmit the data between micro:bits for logging. **CT – Concepts, Practices, Perspectives:** Throughout the project, Mr. Mayer introduces CT concepts like variables, loops, and conditionals. Students practice abstraction by designing functions for data collection and transmission. They gain perspective on system thinking by considering the entire data flow from sensor to central logger. **CS-unplugged:** To illustrate radio communication concepts, Mr. Mayer uses a human chain activity where students pass messages (data packets) across the room, demonstrating concepts like data loss and packet order.

Data Logging Implementation: (1) Students program multiple micro:bits as data collectors: some placed indoors, others outdoors. (2) Each collector micro:bit is programmed to: - Read temperature and humidity at regular intervals (e.g., every 15 minutes) - Store readings temporarily with timestamps - Transmit data packets via radio at set intervals. (3) Students program a central micro:bit as the data logger: - Listen continuously for radio transmissions - Store received data in the micro:bit's memory - Display basic statistics on its LED screen (e.g., daily high/low temperatures). (4) Data Analysis on micro:bit/computer: - Calculate and display average temperatures and humidity levels - Show temperature trends using LED brightness or simple graphs - Compare indoor vs. outdoor readings using scrolling displays - Program the central logger to export data for basic analysis.

Suggested Extension Activities: (1) Students design a simple data visualization for the micro:bit's LED display. For example, they might create a scrolling display of 24-hour temperature changes or use LED brightness to indicate temperature ranges.

(2) They program alerts for extreme temperature or humidity conditions. (3) Students compare data from different locations within the school, investigating factors that might influence temperature and humidity.

This project allows Mr. Mayer to integrate CT principles into a practical, long-term data collection effort using micro:bits and external sensors. Students gain hands-on experience with sensor technology, radio communication, data logging, and basic data analysis, all while exploring real-world environmental patterns in their school and community.

Tables and Figures

Table 1. Coverage of the curriculum's two-dimensional competency framework through Handbook Design Principles – (1) Competence Areas (Section 2.2] and (2) 'Frankfurt-Dreieck' △ Perspectives (Section 2.3). – Own representation by the first author.

Action Areas/ *Frankfurt* △ *Perspectives*	**Orientation**	**Information**	**Communication**	**Production**	**Action**
(T) Technological-medial	'Hello-World', IPO, CS-Unplugged [1, 2, 8]	Physical Computing & AHA!, Open-Ended Learning & Makerspaces [5, 6]	–	Evaluation & Debugging, Pair Programming, Full 3D FW, CS-Unplugged [2, 3, 7, 8]	'Hello-World', IPO, Physical Computing & AHA!, CS-Unplugged [1, 2, 6, 7]
(G) Socio-cultural	Open-Ended Learning & Makerspaces [5]	–	Pair Programming [4]	Evaluation & Debugging, Pair Programming [3, 4]	–
(I) Interaction-oriented	Physical Computing & AHA! [6]	Evaluation & Debugging, Full 3D FW, Open-Ended Learning & Makerspaces [3, 5, 7]	Pair Programming, Open-Ended Learning & Makerspaces [4, 5]	Physical Computing & AHA!, Full 3D FW,[6, 7]	Open-Ended Learning & Makerspaces [5]

Handbook Design Principles: [1] 'Hello World', [2] Input-Process-Output, [3] Evaluation & Debugging, [4] Pair Programming, [5] Open-Ended Learning & Makerspaces, [6] Physical Computing & *AHA!* Experience, [7] Full 3D Framework for CT: Concepts, Practices & Perspectives, [8] CS-Unplugged Activities. **(See Section 4 – Results)**

Figures: No Figures are provided in the Appendix.