Developing Design Principles for Computational Thinking Learning Environments: Pathways into Practice with Physical Computing

Oliver Kastner-Hauler[1][a], Bernhard Standl[2][b], Barbara Sabitzer[3][c] and Zsolt Lavicza[3][d]

[1] *University of Education Lower Austria, Department of Media Education, 2500 Baden, Austria*

[2] *Karlsruhe University of Education, Department of Informatics and Digital Education, 76133 Karlsruhe, Germany*

[3] *Johannes Kepler University Linz, Department of STEM Education, 4040 Linz, Austria*

Keywords: Computational Thinking, Learning Environment, Design Principles, Practical Handbook, Physical Computing, Digital Education.

Abstract: To support K-8 educators in integrating computational thinking (CT) into basic digital education (BDE), as introduced in Austria in 2018, we present design principles for a practical handbook. Teachers without a background in computer science (CS) may hesitate to teach CT and prioritize media and computer literacy aspects of BDE, potentially neglecting CT in the revised 2022 curriculum. To overcome this, we iteratively developed design principles for a handbook with three practical learning environments (LEs) employing physical computing with a single-board computer. The LEs embrace the constructionist approach and emphasize experiential learning to support intrinsic learning of CS/CT concepts. Complementary, an Open Educational Resource (OER) textbook is available that is aligned with the 5E instructional model and promotes self-directed, inquiry-based learning. This makes CS/CT more accessible to both teachers and students, regardless of their CS knowledge. Following these principles, they are systematically guided to gain confidence in applying CT in the classroom. Further research will evaluate and refine the design principles on a larger scale, contributing to the development of a comprehensive handbook. Ensuring accessibility to fundamental CT skills in K-8 education is crucial for the successful implementation of basic digital education (BDE).

1 INTRODUCTION

In recent years, integrating computational thinking (CT) into education has become vital for preparing students for future careers (Fraillon et al., 2020; WE Forum, 2023). CT involves using tools and techniques of computer science (CS) to solve problems across various disciplines, extending beyond CS (Wing, 2006). Basic digital education (BDE) combines CT, media, and computer literacy in Austria. It has evolved from an exercise in 2018 to a subject in 2022 mandatory for grades 5 to 8 (BMBWF, 2022). Introducing a new subject can be challenging for teachers, even with in-service training options. Teachers who do not feel prepared may focus on other aspects of BDE, omitting CT and the foundations of any digital education. This study aims to establish guiding principles for content selection and development towards a practice-oriented handbook, founded on literature and curricula, through a design-based research (DBR) project (McKenney and Reeves, 2013; McKenney and Reeves, 2018). Three learning environments (LEs) that facilitate effective CT education guide teachers and students through their initial steps without the need for prior CS knowledge (Kastner-Hauler et al., 2022; Kastner-Hauler et al., 2021; Brandhofer and Kastner-Hauler, 2020). Teachers can apply the design principles developed from the LEs to create new or enrich existing material for BDE, regardless of their formal background in CS (Hromkovič and Lacher, 2017). By eliminating the need for prior CS knowledge and encouraging joyful and playful practice along the learning environments (LEs), we have found a promising approach to overcome entry barriers to learning and teaching CT. This approach also serves as the missing bridge connecting media education, computer literacy, and informatics (CS, including CT), which is crucial to the successful implementation of digital education in today's schools (Diethelm, 2022).

[a] https://orcid.org/0000-0002-9958-3298
[b] https://orcid.org/0000-0002-8849-2980
[c] https://orcid.org/0000-0002-1304-6863
[d] https://orcid.org/0000-0002-3701-5068

Kastner-Hauler, O., Standl, B., Sabitzer, B. and Lavicza, Z.
Developing Design Principles for Computational Thinking Learning Environments: Pathways into Practice with Physical Computing.
DOI: 10.5220/0012689600003693
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Conference on Computer Supported Education (CSEDU 2024) - Volume 1*, pages 445-453
ISBN: 978-989-758-697-2; ISSN: 2184-5026
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

2 BACKGROUND

Defining computational thinking (CT) is an ongoing process (Selby and Woollard, 2013) that Palts and Pedaste (2020) captured as a visual overview of the many CT definitions over time. Tang et al. (2020) and Weintrop et al. (2021) provide a comparable picture of CT assessments in the field. Li et al. (2020) see CT as a mental model for thinking and problem-solving, rather than as a sole computing skill. Taking this view, we designed our handbook principles to be able to adapt to future advances in CT. The dynamic nature of computing can facilitate the incorporation of CT across STEAM disciplines (Pears et al., 2019). The integration of computational thinking (CT) skills with block-based programming, physical computing, and inquiry-based learning under the umbrella of constructivist learning (Papert, 1980) forms the basis for the principles of the handbook. CT in the educational context is strongly related to programming skills (Brennan and Resnick, 2012) with block-based environments such as Scratch (Scratch Foundation, 2023). CT concepts applied through block-based programming provide a foundation for students to develop their problem-solving skills (Labusch et al., 2019; Nouri et al., 2020). With physical computing, the CT concepts become more understandable as tangible solutions to real-world problems are created and computing is translated into the physical world (Genota, 2019). Physical computing also enables hands-on experience and promotes student-centered and self-directed learning, implemented with the developed inquiry-based learning (IBL) materials (Tkáčová et al., 2019). The IBL materials, including the OER textbook and the wiki, focus on active exploration and learning participation, which are core to the constructivist learning process (Serafín et al., 2015). They emphasize the importance of hands-on learning and social interactions in the creation of the final product (Holbert et al., 2020). Based on this pedagogical background, three learning environments were created, applying the 3D Framework of 4Ps (Brennan and Resnick, 2012) merged with COOL Informatics (Sabitzer et al., 2019). The derived principles of the handbook combine these approaches to provide students and teachers with meaningful and engaging learning experiences that promote CT along with creativity, collaboration, and critical thinking skills. In the following, the building blocks of this research are presented in more detail.

2.1 Block-Based Programming

Text-based programming languages are frequently used for upper school levels and favor those who understand English, while block-based languages are favored for lower school levels because of their lower entry barrier (Weintrop and Wilensky, 2015; Weintrop and Wilensky, 2017). Step-by-step the instructions are put together to code a solution like building a toy house with snap-together construction bricks. The LEs presented in this paper use the official block-based programming environment Makecode for micro:bit (Microbit Foundation, 2016). It has been reported that block-based coding develops problem-solving, planning, and thinking strategies and helps to develop social, language, and cognitive skills, even for early childhood ages (Papadakis, 2022). Papert underlines the importance of programming as an essential component of each individual's intellectual development (Kestenbaum, 2005).

2.2 Physical Computing

Physical computing connects a computing device to the environment to sense and communicate, allowing learners to design and create tangible real-world products (Cápay and Klimová, 2019). The artifacts reproduced from memory manifest the mental concepts of the thinking process during problem-solving (Papert and Harel, 1991). The physical component, therefore, enhances the understanding of programming concepts and expands the imagination of what can be accomplished with code (Przybylla and Romeike, 2014).

2.3 Inquiry-Based Learning

Inquiry-based learning (IBL), anchored by the 5E instructional model – engage, explore, explain, elaborate, and evaluate – guides the handbook's LEs (Pedaste et al., 2015). The model gives teachers the flexibility to implement various questioning strategies, from open-ended to direct, thus tailoring to diverse student needs. The learning materials are specifically designed to encourage playful exploration and problem-solving, nurturing students' curiosity for deeper investigation (Ah-Nam and Osman, 2017). Through interactive elements on the wiki, learners start with a subset of the necessary information for a sample exercise, with additional resources hidden behind spoiler links. This blend of guided and self-directed discovery promotes inquiry and active investigation (Guzdial, 2015).

The background presented is inspired by the 8 Big Ideas of Papert's Constructionist Learning Lab (Martinez, 2017), which emphasizes learning-by-doing and active student participation in knowledge construction (Wagh et al., 2017). His ideas serve as a guiding mantra for the subsequent methods employed to answer the following research question.

RQ: What essential design principles should inform the development of a practical K-8 handbook for computational thinking (CT) learning environments (LEs), with ongoing refinement through design-based research (DBR) to enhance teacher confidence and efficacy in integrating CT into basic digital education (BDE), especially for non-specialists in computer science (CS)?

3 METHODS: DEVELOPMENT OF HANDBOOK PRINCIPLES

With this study, we present an important milestone of a comprehensive design-based research (DBR) project (McKenney and Reeves, 2013; McKenney and Reeves, 2018) aimed at developing principles for a practical handbook. Initiated in 2019, this project introduced the micro:bit and computational thinking (CT) concepts to selected primary and secondary schools, focusing on teachers and students aged 8 to 14. The milestone comprises three investigations: adapting an OER for flipped learning (Kastner-Hauler et al., 2021), examining combined effects of physical computing and block-based programming in primary school (Kastner-Hauler et al., 2022), and assessing CT teaching efficacy in middle school with Bebras (2023) – the latter is currently under review. The participating classroom teachers received specialized training to implement the LEs and assessment tools effectively.

To enhance the balance between digital media, computer literacy, and informatics (CS/CT) for the new subject BDE, we investigated programming in the context of media use and creation. Early programming education can be pursued with platforms like Scratch from the MIT Media Lab (Resnick, 2014) or Makecode (Microsoft, 2024). Resnick and Rusk (2020) highlight 4 pillars of effective digital education – projects, passion, peers, and play. The 4Ps advocate for engaging students in meaningful projects, nurturing passion, fostering collaboration, and playful experimentation. Similar aspects are described with the COOL Informatics approach (Sabitzer et al., 2019) and its 4 principles – discovery, cooperation, individuality, and activity. COOL Informatics incorporates the neurodidactical learning perspective, which we view as an extension of the 4Ps. Merging these approaches, we developed three learning environments (LEs) for CT, underpinning the principles for a practical handbook dedicated to actionable classroom strategies. Our analysis showed a synergy between 4Ps and COOL Informatics, particularly in the categories 'peers' of 4Ps and 'cooperation' of COOL Informatics (see Appendix - Table 2). As this paper focuses more on programming, we present the 4Ps in their practical implementation in Table 1.

The principles derived encompass all aspects of the 4Ps, progressing logically from theory to practical application. This framework guides orientation, enhances understanding, and raises awareness of CT in the classroom. By adhering to the principles outlined in the following section, a comprehensive connection is established across all aspects of basic digital education (BDE), including CT, media, and computer literacy.

4 RESULTS: HANDBOOK DESIGN PRINCIPLES

This outline presents principles from design research for creating engaging and streamlined learning environments (LEs) for coding and problem-solving. It offers a guide for teachers, particularly those new to computer science (CS), to integrate CT into BDE lessons. The guide covers the essential aspects of 4Ps and COOL Informatics, facilitating classroom implementation without unnecessary complexity. By adhering to these principles, teachers without a CS background can effectively prepare for the new BDE subject, ensuring seamless lesson planning and teaching.

4.1 'Hello World'

Asking *why* and *what for* builds programming understanding, but getting into action quickly is key. The first task in learning a new programming language usually involves outputting 'Hello World' on the display. Using the Makecode editor and the micro:bit, learners can easily begin, focusing on Makecode first and its integrated online simulator. Beginners start making a heart appear on the 5 x 5 LED display instead of 'Hello World'. The programming blocks, organized into color-coded function groups, are easy to find and use. Learners can create programs playfully by dragging and dropping blocks of code, experimenting until they achieve the desired outcome.

4.2 Input-Process-Output

The Input-Process-Output (IPO) principle illustrates the computer's mode of operation. After demonstrat-

Table 1: Implementing 4Ps of Creative Learning, (Resnick, 2014) – A Roadmap for the Natural and Logical Progression of CT Development to Identify Action Areas for the Design Principles of a Practical Handbook (short version).

concepts/ *methods*	**projects**	**passion**	**peers**	**play**
block-based programming	foster imagination, sharing of final code [1, 2, 7]	story evokes interest, immediate activity [2, 5, 6]	pair programming artifacts, textbook wiki [3, 4, 7]	self-directed, individual, experiental learning [1, 3, 6]
physical computing	tangible products with sensors and actuators [2, 5, 6]	foster open-ended making possibilities [3, 5, 6]	physical artifacts ready to share [4, 5, 6]	self-directed, individual, hands-on learning [5, 6, 8]
inquiry-based learning using 5E cycle	promotes a cycle of permanent improvement [3, 5, 6]	strengthens individual desire to improve [3, 6, 7]	lively exchange stimulates new knowledge [4, 6, 8]	self-directed, individual, incremental learning [5, 6, 7]

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] CT: Concepts, Practices & Perspectives, [8] CS-Unplugged Activities. **(See Section 4 Results)**

ing the output with the heart, we introduce the input with micro:bit's A and B buttons. Pressing A to display the heart and B to clear mimics a flashing heart when alternately pressed. Highlighting the importance of automated (data) processing, we use traffic light's 24/7 operation as an analogy. To automate the switching and flashing, we employ automatic processing of the buttons via a loop. Pressing A+B simultaneously starts the process on the micro:bit and the display switches from heart to blank. The result is an automated *flashing heart*, ready for subsequent evaluation and testing.

4.3 Evaluation and Debugging

Initial testing will likely show that the blinking works once with A+B triggers, emphasizing the important principle of evaluation in programming. If programming outcomes do not meet intentions, debugging is essential to identify and correct flaws in code (Kim et al., 2018). Makecode's step-by-step debugger and verbalizing the program's flow and actions can aid the process (Heikkilä and Mannila, 2018). Encouraging students to explain their code to peers (Tengler et al., 2022), focusing on what exactly the code does and its sequence, enhances understanding. Pair programming principles should be applied to evaluation and debugging, and broadly across the whole thinking process during coding.

4.4 Pair Programming

Budget constraints can often make it impractical to provide one device per child in new projects. Pair programming offers a solution and educational benefits, where two programmers create a joint solution on a single device. One acts as the driver, typing the code, while the other guides the process as the navigator (Bryant et al., 2008). This setup, similar to the metaphor of driving a car, ensures clear communication and understanding. Recommended regular role reversal improves code quality and peer interaction (Graßl and Fraser, 2023). To enrich the interaction, it is recommended to apply peer learning and teaching (Kröhn and Sabitzer, 2020) with the open-ended learning tasks provided in the textbook wiki. Additionally, animated tutorials from Makecode can also be beneficial, serving as either peer substitute or third-person perspective for code demonstration.

4.5 Open-Ended Learning & Makerspaces

The OER textbook promotes open-ended learning, encouraging students to tackle problems with various solutions, and leveraging creativity, collaboration, and critical thinking skills in the context of CT (Tsan et al., 2022). The textbook suggests the use of sample extensions for further exploration and development of micro:bit projects. Practical experiences, through makerspaces, games, or real-world challenges, can deepen knowledge. Makerspace activities, suitable for flipped classroom delivery, split into programming (pre-class) and making (in-class) maximize hands-on time (Kastner-Hauler et al., 2021). Open-ended learning fosters a growth mindset, a sense of ownership of learning, and the discovery of new opportunities through successes and challenges in activities involving physical computing.

4.6 Physical Computing & *AHA!* Experience

Connecting a micro:bit to a computer provides a haptic experience of the code and algorithms created (Kalelioglu and Sentance, 2020). Online tutorials available on the textbook wiki or the Makecode website offer guidance for initial physical steps. After mastering code upload, a portable power supply becomes necessary. This marks the first *AHA!* moment (Thagard and Stewart, 2011) as concepts materialize in real life through the device's haptic aspect (Spiridonov et al., 2019). Equipped with onboard sensors and actuators, the micro:bit enables interaction with the physical world, exemplified by a portable step counter (Microbit Foundation, 2023). For more *AHA!* moments and creative exploration with physical computing, attach the micro:bit to a person's leg and count the steps using the accelerometer (O'Sullivan and Igoe, 2004).

4.7 CT Dimensions: Concepts, Practices and Perspectives

Full awareness of all dimensions of Brennan and Resnick's (2012) 3D framework equips educators to design and evaluate CT learning experiences for diverse students. **CT concepts**, like sequences, loops, conditionals, data, and operators, are foundational to coding and can be assessed with the validated Beginners Computational Thinking test (BCTt) (Zapata-Cáceres et al., 2021). Assessments of CT were investigated in previous research cycles (Kastner-Hauler et al., 2022; Brandhofer and Kastner-Hauler, 2020) and are not included in this paper, focusing on the handbook principles. **CT practices and perspectives** – how concepts are applied and the attitudes towards computing – cannot be directly measured from programming artifacts produced with the Makecode environment for micro:bit. The tool Dr. Scratch (Moreno-León et al., 2015), which assesses CT in Scratch programs, suggests potential for a Makecode equivalent. Research into creating a Dr. Microbit tool for analyzing micro:bit projects could fill this gap. Meanwhile, we sought to identify CT practices interwoven with the LEs' material, such as evaluation and debugging, and CT perspectives through feedback on pair programming and CS-unplugged activities. By evaluating the LEs as a whole, including feedback, we indirectly cover CT practices and perspectives.

4.8 CS-Unplugged Activities

Utilize CS-Unplugged activities to teach computer science (CS) fundamentals without the use of computers (Bell and Vahrenhold, 2018). This free resource uses games and tasks with common items like cards and crayons to demystify CS concepts, making them accessible for learners of all ages. These activities match efforts from similar projects by ADA (VCLA, 2023) and COOL Lab (JKU, 2023). For example, a demonstration of the parity magic trick can be a fun way to learn more about error detection and correction, a concept crucial for data transmission. CS-unplugged presents a unique, hands-on approach to grasp foundational CS and CT principles, vital for any digital education (Curzon and McOwan, 2018; Rottenhofer et al., 2022).

5 DISCUSSION & CONCLUSION

Computational thinking (CT) is reflected in the technological-medial aspect of the Frankfurt Triangle (Brinda et al., 2019) on digitization in education outlined in the BDE curriculum (BMBWF, 2022), without direct mention. The lack of explicitness can challenge teachers' readiness to incorporate CT in BDE lessons, despite optional in-service training. The approach presented here advocates for a gentle introduction to computer science (CS) and computational thinking (CT) by developing design principles for a practical handbook, drawing from ongoing educational design research (EDR) (Bakker, 2018; McKenney and Reeves, 2013; Plomp and Nieveen, 2013). This approach makes CT more accessible and actionable for teachers new to the BDE subject, enhancing their confidence and facilitating learning and understanding of basic CS concepts and CT skills through learning-by-doing.

The design principles that emerged emphasize core programming concepts such as 'Hello World', Input-Process-Output, and debugging with collaborative techniques like pair programming. These foundational practices encourage a problem-solving mindset and require reflective thinking and initial guidance, supported by additional scaffolding material (Tsan et al., 2022). Incorporating open-ended challenges (Emara et al., 2021) in combination with makerspaces and physical computing into teaching necessitates precise guidelines for effective implementation. The holistic view of CT concepts, principles, and perspectives brings all dimensions together and makes them accessible. This concept is embedded in all the material and can be highlighted through CS-unplugged activities without using computers. An example in the Appendix demonstrates how these principles can be applied in practice.

Drawing on Hsu's (2018) comprehensive review, our design principles for a handbook on computational thinking (CT) are based on effective learning and teaching strategies and best practice knowledge. This offers a solid foundation for educators who seek to integrate CT into their classrooms (Csizmadia et al., 2019), regardless of a formal background in CS (Hromkovič and Lacher, 2017).

Further research is planned to evaluate these principles through a teacher training course on basic digital education (BDE), focusing on the impact of the principles on the acceptance of CT with BDE for future incorporation into one's teaching. Furthermore, examining how teachers adapt and create learning materials based on these principles will lead to a solid support handbook, refining the content and design through iterative reflections.

The utilization of computational thinking (CT) in classrooms is a crucial opportunity for its further development, supported by gradual accumulation of knowledge through experiential learning. By embedding computational thinking (CT) constructs such as algorithms, abstraction, and automation, students and teachers can deepen their understanding of fundamental computer science (CS) principles and become computational thinkers (Yadav et al., 2016). As students and teachers engage with computational thinking (CT) and apply its problem-solving habits, they naturally develop confidence and fluency, empowering them to tackle the complex challenges of the unknown future with ease.

REFERENCES

Ah-Nam, L. and Osman, K. (2017). Developing 21st century skills through a constructivist-constructionist learning environment. *K-12 Stem Education*, 3(2):205–216. https://www.learntechlib.org/p/209542/, [Accessed:2023-03-15].

Bakker, A. (2018). *Design research in education: A practical guide for early career researchers.* Routledge.

Bebras (2023). What is Bebras International Challenge on Informatics and Computational Thinking. https://www.bebras.org/about.html, [Accesed:2023-05-12].

Bell, T. and Vahrenhold, J. (2018). CS Unplugged - How Is It Used, and Does It Work? In *Adventures Between Lower Bounds and Higher Altitudes.* Springer.

BMBWF (2022). *Lehrplan Digitale Grundbildung*. Bundesministerium für Bildung Wissenschaft und Forschung. https://www.ris.bka.gv.at/Dokumente/BgblAuth/BGBLA_2022_II_267/BGBLA_2022_II_267.pdfsig, [Accessed:2023-01-01].

Brandhofer, G. and Kastner-Hauler, O. (2020). Der micro:bit und Computational Thinking. Evaluierungsergebnisse zu einem informatischen Projekt. *R&E-SOURCE.* https://journal.ph-noe.ac.at/index.php/resource/article/view/914, [Accessed:2023-09-15.

Brennan, K. and Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association (AERA), Vancouver, Canada*, volume 1, page 25.

Brinda, T., Brüggen, N., Diethelm, I., Knaus, T., Kommer, S., Kopf, C., Missomelius, P., Leschke, R., Tilemann, F., and Weich, A. (2019). Frankfurt-Dreieck zur Bildung in der digital vernetzten Welt. *Informatik für alle, Gesellschaft für Informatik.*

Bryant, S., Romero, P., and Du Boulay, B. (2008). Pair programming and the mysterious role of the navigator. *International Journal of Human-Computer Studies*, 66(7):519–529.

Csizmadia, A., Standl, B., and Waite, J. (2019). Integrating the Constructionist Learning Theory with Computational Thinking Classroom Activities. *Informatics in Education*, 18(1):41–67.

Curzon, P. and McOwan, P. W. (2018). *Computational Thinking: Die Welt des algorithmischen Denkens – in Spielen, Zaubertricks und Rätseln.* Springer Berlin Heidelberg.

Cápay, M. and Klimová, N. (2019). Engage Your Students via Physical Computing! In *2019 IEEE Global Engineering Education Conference EDUCON)*, pages 1216–1223.

Diethelm, I. (2022). Digital Education and Informatics – You can't have One without the Other. In *Proceedings of the 17th Workshop in Primary and Secondary Computing Education*, pages 1–2. ACM.

Emara, M., Hutchins, N., Grover, S., Snyder, C., and Biswas, G. (2021). Examining Student Regulation of Collaborative, Computational, Problem-Solving Processes in Open-Ended Learning Environments. *Journal of Learning Analytics*, 8(1):49–74.

Fraillon, J., Ainley, J., Schulz, W., Friedman, T., and Duckworth, D. (2020). *Preparing for Life in a Digital World: IEA International Computer and Information Literacy Study 2018 International Report*. Springer International Publishing.

Genota, L. (2019). 'Physical Computing' Connects Computer Science With Hands-On Learning. *Education Week.* https://www.edweek.org/teaching-learning/physical-computing-connects-computer-science-with-hands-on-learning/2019/01, [Accessed 2023-05-27].

Graßl, I. and Fraser, G. (2023). The ABC of Pair Programming: Gender-dependent Attitude, Behavior and Code of Young Learners. In *ICSE 2023 Proceedings*, page 13.

Guzdial, M. (2015). *Learner-Centered Design of Computing Education: Research on Computing for Everyone*. Synthesis Lectures on Human-Centered Informatics. Springer.

Heikkilä, M. and Mannila, L. (2018). Debugging in Programming as a Multimodal Practice in Early Childhood Education Settings. *Multimodal Technologies and Interaction*, 2(3):42.

Holbert, N., Berland, M., and Kafai, Y. B., editors (2020). *Designing constructionist futures: the art, theory, and practice of learning designs*. The MIT Press.

Hromkovič, J. and Lacher, R. (2017). 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.

Hsu, T.-C., Chang, S.-C., and Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, 126:296–310.

JKU (2023). COOL Lab, Johannes Kepler Universität Linz. https://www.jku.at/en/schools/cool-lab/, [Accessed:2023-06-04].

Kalelioglu, F. and Sentance, S. (2020). Teaching with physical computing in school: the case of the micro:bit. *Education and Information Technologies*, 25(4):2577–2603.

Kastner-Hauler, O., Tengler, K., Demarle-Meusel, H., and Sabitzer, B. (2021). Adapting an OER textbook for the inverted classroom model - how to flip the classroom with BBC micro:bit example tasks. In *2021 IEEE Frontiers in Education Conference (FIE)*, pages 1–8. IEEE.

Kastner-Hauler, O., Tengler, K., Sabitzer, B., and Lavicza, Z. (2022). Combined effects of block-based programming and physical computing on primary students' computational thinking skills. *Frontiers in Psychology*, 13:875382.

Kestenbaum, D. (2005). The challenges of IDC: what have we learned from our past? *Communications of the ACM*, 48(1):35–38.

Kim, C., Yuan, J., Vasconcelos, L., Shin, M., and Hill, R. B. (2018). Debugging during block-based programming. *Instructional Science*, 46(5):767–787.

Kröhn, C. and Sabitzer, B. (2020). Peer-learning and Talents Exchange in Programming: Experiences and Challenges:. In *Proceedings CSEDU 2020*, pages 466–471. SCITEPRESS.

Labusch, A., Eickelmann, B., and Vennemann, M. (2019). Computational Thinking Processes and Their Congruence with Problem-Solving and Information Processing. In *Computational Thinking Education*, pages 65–78. Springer.

Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., and Duschl, R. A. (2020). Computational Thinking Is More about Thinking than Computing. *Journal for STEM Education Research*, 3(1):1–18.

Martinez, S. (2017). Around the World with the 8 Big Ideas of the Constructionist Learning Lab. https://inventtolearn.com/around-the-world-with-the-8-big-ideas-of-the-constructionist-learning-lab/, [Accessed 2023-05-29].

McKenney, S. and Reeves, T. C. (2013). Educational Design Research. In *Handbook of Research on Educational Communications and Technology*, pages 131–140. Springer.

McKenney, S. and Reeves, T. C. (2018). *Conducting educational design research*. Routledge, 2nd edition.

Microbit Foundation (2016). The Micro:bit Educational Foundation. https://microbit.org/about/, [Accessed 2024-12-20].

Microbit Foundation (2023). micro:bit step counter. https://microbit.org/projects/make-it-code-it/step-counter/, [Accessed:2023-08-04].

Microsoft (2024). Microsoft MakeCode for micro:bit. https://makecode.microbit.org/, [Accessed 2024-11-11].

Moreno-León, J., Robles, G., and Román-González, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED. Revista de Educación a Distancia*, 46(1):1–23.

Nouri, J., Zhang, L., Mannila, L., and Norén, E. (2020). Development of computational thinking, digital competence and 21 st century skills when learning programming in K-9. *Education Inquiry*, 11(1):1–17.

O'Sullivan, D. and Igoe, T. (2004). *Physical computing: sensing and controlling the physical world with computers*. Thomson.

Palts, T. and Pedaste, M. (2020). A Model for Developing Computational Thinking Skills. *Informatics in Education*, 19(1):113–128.

Papadakis, Stamatios, P. (2022). Can Preschoolers Learn Computational Thinking and Coding Skills with ScratchJr? A Systematic Literature Review. *International Journal of Educational Reform*, pages 1–34.

Papert, S. (1980). *Mindstorms; Children, Computers and Powerful Ideas*. Basic Books.

Papert, S. and Harel, I. (1991). Situating constructionism. *Constructionism*, 36(2):1–11.

Pears, A., Barendsen, E., Dagienė, V., Dolgopolovas, V., and Jasutė, E. (2019). Holistic STEAM Education Through Computational Thinking: A Perspective on Training Future Teachers. In *Informatics in Schools. New Ideas in School Informatics*, volume 11913, pages 41–52. Springer International Publishing.

Pedaste, M., Mäeots, M., Siiman, L. A., de Jong, T., van Riesen, S. A., Kamp, E. T., Manoli, C. C., Zacharia, Z. C., and Tsourlidaki, E. (2015). Phases of inquiry-based learning: Definitions and the inquiry cycle. *Educational Research Review*, 14:47–61.

Plomp, T. and Nieveen, N., editors (2013). *Educational design research.*, volume Part A: An introduction. Netherlands Institute for Curriculum Development (SLO).

Przybylla, M. and Romeike, R. (2014). Physical Computing and its Scope - Towards a Constructionist Computer Science Curriculum with Physical Computing. *Informatics in Education*, 13(2):225–240.

Resnick, M. (2014). Give P's a chance: Projects, peers, passion, play. In *Constructionism and creativity: Proceedings of the third international constructionism conference. Austrian computer society, Vienna*, pages

13–20. https://www.media.mit.edu/~mres/papers/constructionism-2014.pdf, [Accessed:2024-01-22].

Resnick, M. and Rusk, N. (2020). Coding at a crossroads. *Communications of the ACM*, 63(11):120–127.

Rottenhofer, M., Kuka, L., and Sabitzer, B. (2022). Clear the Ring for Computer Science: A Creative Introduction for Primary Schools. In *Informatics in Schools. ISSEP 2022*, volume 13488, pages 103–112. Springer International Publishing.

Sabitzer, B., Demarle-Meusel, H., and Painer, C. (2019). A COOL Lab for Teacher Education. In *Rethinking Teacher Education for the 21st Century: Trends, Challenges and New Directions*, pages 319–328. Verlag Barbara Budrich.

Scratch Foundation (2023). Scratch - About. https://scratch.mit.edu/about, [Accessed 2023-08-04].

Selby, C. and Woollard, J. (2013). Computational Thinking: The Developing Definition. https://eprints.soton.ac.uk/356481/1/Selby_Woollard_bg_soton_eprints.pdf, [Accessed 2023-05-12].

Serafín, Č., Dostál, J., and Havelka, M. (2015). Inquiry-Based Instruction in the Context of Constructivism. *Procedia - Social and Behavioral Sciences*, 186:592–599.

Spiridonov, V., Loginov, N., Ivanchei, I., and Kurgansky, A. V. (2019). The Role of Motor Activity in Insight Problem Solving (the Case of the Nine-Dot Problem). *Frontiers in Psychology*, 10:1–17.

Tang, X., Yin, Y., Lin, Q., Hadad, R., and Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, 148:103798.

Tengler, K., Kastner-Hauler, O., Sabitzer, B., and Lavicza, Z. (2022). The Effect of Robotics-Based Storytelling Activities on Primary School Students' Computational Thinking. *Education Sciences*, 12(1):1–10.

Thagard, P. and Stewart, T. C. (2011). The AHA! Experience: Creativity Through Emergent Binding in Neural Networks. *Cognitive Science*, 35(1):1–33.

Tkáčová, Z., Šnajder, L., and Guniš, J. (2019). Inquiry-Based Learning in Computer Science Classroom. In Pozdniakov, S. N. and Dagienė, V., editors, *Informatics in Schools. ISSEP 2019*, volume 11913, pages 68–79. Springer.

Tsan, J., Eatinger, D., Pugnali, A., Gonzalez-Maldonado, D., Franklin, D., and Weintrop, D. (2022). Scaffolding Young Learners' Open-Ended Programming Projects with Planning Sheets. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 1*, pages 372–378. ACM.

VCLA (2023). CS Unplugged – Materialiensammlung, Vienna Center for Logic and Algorithms. https://www.ada.wien/cs-unplugged-materialiensammlung/, [Accessed:2023-06-04].

Wagh, A., Cook-Whitt, K., and Wilensky, U. (2017). Bridging inquiry-based science and constructionism: Exploring the alignment between students tinkering with code of computational models and goals of inquiry. *Journal of Research in Science Teaching*, 54(5):615–641.

WE Forum (2023). *The Future of Jobs Report 2023*. World Economic Forum. https://www.weforum.org/reports/the-future-of-jobs-report-2023/, [Accessed 2023-05-12].

Weintrop, D. and Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*, pages 199–208. ACM.

Weintrop, D. and Wilensky, U. (2017). Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. *ACM Transactions on Computing Education*, 18(1):1–25.

Weintrop, D., Wise Rutstein, D., Bienkowski, M., and McGee, S. (2021). Assessing computational thinking: an overview of the field. *Computer Science Education*, 31(2):113–116.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3):33–35.

Yadav, A., Hong, H., and Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends*, 60:565–568.

Zapata-Cáceres, M., Martín-Barroso, E., and Román-González, M. (2021). BCTt: Beginners Computational Thinking Test. In *Understanding computing education*, Proceedings of the Raspberry Pi Foundation Research Seminar series, pages 46–56. Raspberry Pi Foundation.

APPENDIX

Example of Applied Handbook Principles

Mrs. Jane Doe, a middle school teacher, wants to integrate computational thinking (CT) into her curriculum. She doesn't have a formal background in computer science (CS), but she recognizes the significance of CT skills for the future of her students. Mrs. Doe decided to use the principles of the handbook as presented in this paper to create a new project for her students to work on. The project involves the collection and analysis of environmental data and responses using a single-board computing device such as the here used micro:bit.

'Hello World': Mrs. Doe begins by asking the students to create a simple program that displays a message or symbol, such as a smiley face, on the micro:bit's LED screen. This gives her students a quick win and familiarizes them with the programming environment and features of the micro:bit. **Input-Process-Output:** Next, Mrs. Doe explains the concept of input-process-output using the micro:bit. Students instruct the micro:bit to collect data from its built-in temperature sensor (input), process the data for conversion to Fahrenheit (process), and display the result on the LED screen (output). **Debugging & Pair Programming:** Mrs. Doe assigns students to work in pairs to create and test their programs. She encourages the use of pair programming, where one student writes the code, and the other student provides instructions and feedback. She explains that this improves the quality of the program and helps them learn from each other. **Open-ended Learning, Makerspaces & Inquiry-Based Learning:** Mrs. Doe then sets an open-ended challenge for the students. She also emphasizes the use of the textbook wiki for self-directed, inquiry-based learning. Students have to design and build a device using the micro:bit that can help solve an environmental problem e.g. watering a plant. The students work in the school's makerspace, using various materials and tools to build their devices. **Physical Computing & *AHA!* Experience:** Students experience *AHA!* moments that come from seeing their code interact with the physical world as they work on their projects. For example, students can program their device to alert them when soil moisture drops below a certain level to help water a plant. **CT – Concepts, Practices, Perspectives:** Throughout the project, Mrs. Doe uses the 3D framework for CT intertwined with COOL Informatics to guide her teaching and assess the learning of her students. She helps students understand fundamental concepts of CT, such as loops and conditionals and encourages them to reflect on their learning process. **CS-unplugged:** Mrs. Doe uses CS-unplugged activities throughout the whole project where suitable to illustrate CS concepts. For example, when programming plant watering, certain conditions must be checked. Conditionals can be demonstrated by playing the Simon Says game if-then-else. In this way, the CS fundamentals are learned in a fun and engaging way.

In addition to helping Mrs. Doe integrate CT into her curriculum, this project provides a hands-on, engaging learning experience for her students. As they apply their knowledge to solve real-world problems, they learn important CT skills.

Combined 4Ps and COOL Informatics

Table 2: A Roadmap for the Natural and Logical Progression of CT Development to Identify Action Areas for the Design Principles of a Practical Handbook (extended version.

	4Ps of Creative Learning, Resnick (2014) Scratch block-based programming, media use and creation				**COOL Informatics *, Sabitzer et al. (2019)** Computer Science unplugged aspects			
concepts/ *methods*	**projects**	**passion**	**peers**	**play**	**discovery**	**cooperation**	**individuality**	**activity**
block-based programming	foster imagination, sharing of final code [1, 2, 7]	story evokes interest, immediate activity [2, 5, 6]	pair programming artifacts, textbook wiki [3, 4, 7]	self-directed, individual, experiental learning [1, 3, 6]	immediate activity, making extensions [1, 2, 5]	pair programming artifacts, textbook wiki [3, 4, 7]	self-directed, individual, holistic learning [2, 6, 7]	immediate activity, holistic learning [6, 7, 8]
physical computing	tangible products with sensors and actuators [2, 5, 6]	foster open-ended making possibilities [3, 5, 6]	physical artifacts ready to share [4, 5, 6]	self-directed, individual, hands-on learning [5, 6, 8]	haptic aspects, self-directed activity foster the spirit of discovery [2, 5, 6]	physical artifacts ready to share [4, 5, 6]	self-directed, individual, experiental learning [3, 5, 6]	haptic aspects, self-directed activity [5, 6, 8]
inquiry-based learning using 5E cycle	promotes a cycle of permanent improvement [3, 5, 6]	strengthens individual desire to improve [3, 6, 7]	lively exchange stimulates new knowledge [4, 6, 8]	self-directed, individual, incremental learning [5, 6, 7]	5E cycle fosters the spirit of discovery [3, 5, 6]	lively exchange stimulates new knowledge [4, 6, 8]	self-directed, individual, incremental learning [1, 2, 5]	5E cycle fosters self-directed, iterative learning [3, 4, 6]

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] CT: Concepts, Practices & Perspectives, [8] CS-Unplugged Activities. **(See Section4 Results)**;

** Focusing on the teaching and learning methods, the neurodidactical dimensions of COOL Informatics are not explicitly shown.*