



Collaborative learning with block-based programming: investigating human-centered artificial intelligence in education

Renate Andersen, Anders I. Mørch & Kristina Torine Litherland

To cite this article: Renate Andersen, Anders I. Mørch & Kristina Torine Litherland (2022): Collaborative learning with block-based programming: investigating human-centered artificial intelligence in education, Behaviour & Information Technology, DOI: [10.1080/0144929X.2022.2083981](https://doi.org/10.1080/0144929X.2022.2083981)

To link to this article: <https://doi.org/10.1080/0144929X.2022.2083981>



© 2022 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group



Published online: 01 Jun 2022.



Submit your article to this journal [↗](#)



Article views: 223



View related articles [↗](#)



View Crossmark data [↗](#)

Collaborative learning with block-based programming: investigating human-centered artificial intelligence in education

Renate Andersen ^a, Anders I. Mørch ^b and Kristina Torine Litherland^b

^aDept. of Primary and Secondary Teacher Education, Oslo Metropolitan University, Oslo, Norway; ^bDepartment of Education, University of Oslo, Oslo, Norway

ABSTRACT

In this article, we investigate human-centered artificial intelligence (HCAI) in an educational context where pupils used block-based programming in small groups to solve tasks given by the teacher. We used a design-based research approach in which we, together with the teachers, created a maker space for explorative science learning and organised teaching interventions wherein the pupils met online three hours a week for 16 weeks for an entire school year. Due to COVID-19, data were collected through Zoom, with collaborative learning situations captured through screen sharing and online communication using webcams. We employed three data analysis techniques: interaction analysis, visual artifact analysis, and thematic analysis. We developed an analytical framework for integration using thematic coding that combined concepts from computer-supported collaborative learning (CSCL) and domain-oriented design environments. We report the following findings: 1) Three types of rules between design units were identified with visual artifact analysis: latent, generic, and domain-specific rules; 2) two types of CSCL artifacts (technology and discussions) were intertwined and developed in parallel, along with a computer-based scaffolding scenario that offloads domain-specific scaffolding from humans to computers.

ARTICLE HISTORY

Received 15 December 2021
Accepted 25 May 2022

KEYWORDS



Block-based programming; computer-supported collaborative learning; domain-oriented design environment; human-centered artificial intelligence; knowledge-based rules; physical computing in education

1. Introduction

Markoff (2016) wrote a popular book about the complicated and evolving relations between humans and computers since the birth of artificial intelligence (AI) in the 1950s to today's embedded computers. In the book, Markoff asks how we should balance what computers can do for us (AI) and what they can help us do ourselves (intelligent augmentation [IA]). Google automated driving and the Apple personal assistant (Siri) are examples of AI and IA, respectively. Our focus is on the IA end of the spectrum. We postulate that knowledge-based rules can support educational researchers to conduct empirical analyses, while IA software can help learners reach their educational goals by reducing the burden for teachers. According to Yang et al. (2021) public opinion about AI is changing from predominantly technology-oriented to humanity-oriented applications. This view is echoed by the United Nations Educational, Scientific and Cultural Organization (UNESCO) (2022), whose mandate is to focus on human-centered approaches to AI. UNESCO (2022) identified three main areas of AI and education as particularly relevant: 1) learning with AI (e.g. the use of AI-empowered tools in classrooms), 2) learning about AI

(AI technologies and techniques), and 3) preparing for AI (e.g. enabling citizens to better understand the potential impact of AI on human lives). Our focus is on learning with AI. In this section, we describe the related work on which our study is based: 1) computer-supported collaborative learning, 2) block-based programming, 3) physical computing in education, 4) end-user development and domain-oriented design environments, and 5) human-centered AI in education. Each of these fields are relevant as they complement each other and in total provides a rich perspective for exploring processes of collaborative learning in a programming context and how it can be explored in the light of HCAI in education.

Due to space limitations, the literature review does not examine the research fields but rather focuses on studies at the intersection of at least two of these fields. We used Google Scholar to identify central articles. Using snowballing, we screened for relevant references in the initial articles and repeated the process. To limit the number of articles, we narrowed our inquiry to the most recently published articles. In addition, we included classics—the canons—which motivated the direction of our work. When searching

CONTACT Renate Andersen  renatea@oslomet.no  Dept. of Primary and Secondary Teacher Education, Oslo Metropolitan University, Oslo, Norway

© 2022 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group

This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (<http://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited, and is not altered, transformed, or built upon in any way.

in the journals and proceedings we used a combination of the keywords listed above, including ‘block-based programming in education,’ ‘computer-supported collaborative learning,’ ‘smart things creation in education,’ and ‘physical computing in education.’ The literature review is not exhaustive but offers insight into the current research landscape, especially regarding HCAI in education in connection with end-user development.

1.1. Computer-supported collaborative learning

CSCL is a research field focused on understanding how people learn together using digital tools, following the assumption that these tools improve the quality of the learning process (Stahl, Koschmann, and Suthers 2006). CSCL researchers have explored how computers can enable students to learn collaboratively in small groups, leading them to share knowledge (Stahl 2006) as a step toward individual learning and achievement at the group interaction level. Artifacts are central in CSCL and play several roles in CSCL settings:

1. They can be technological artifacts supporting communication and collaboration,
2. They can structure the representations that groups of students use when creating shared understandings,
3. They can be instructional artifacts (e.g. teaching models, inquiry scaffolds like sentence openers, and scripts),
4. They can be the result of group efforts oriented toward the co-construction of an artifact, such as a visual artifact, technology design, or project report, and
5. The process might result in a knowledge artifact (e.g. a learning design or a roleplay video) (Stahl et al. 2014).

In our research, we focus on two types of CSCL artifacts that are the result of group efforts, which we refer to as technological and discursive objects. We use the terms visual artifact instead of technology object in some situations. The double naming is due to our dual focus on the empirical analysis of technological objects and a method for analysis of those objects (visual artifact analysis). The terms artifacts and objects are also used interchangeably.

Research methods from the social sciences are commonly used in CSCL to capture group interaction. One common method is interaction analysis (see Methods section). A goal in CSCL is to understand collaborative knowledge construction as it develops from multiple individuals’ personal understandings to a shared understanding through information sharing, positioning, and negotiating multiple perspectives and

interests. Following Stahl (2006), we use the terms shared perspective or group cognition in referring to collaboratively constructed knowledge.

In CSCL contexts, learners can be either physically co-located or distributed in an online learning environment, but learning is always developed through the collaborative creation of shared artifacts (Lipponen, Hakkarainen, and Paavola 2004). Therefore, CSCL is understood as a process involving 1) learner interaction, 2) information sharing, 3) joint meaning making, and 4) the creation of shared artifacts (Andersen 2019; Stahl, Koschmann, and Suthers 2006). In this study, we employ CSCL as a perspective to focus on small group learning in schools, where students are active creators of visual artifacts (physical and digital). The term ‘mutual development’ refers to a process in which different stakeholders collaboratively co-create shared artifacts (Andersen 2019; Andersen and Mørch 2009), like the participants in our intervention. Scaffolding is a key concept in CSCL (Rienties et al. 2012). Vogel et al. (2022) explored how adaptable scaffolding of mathematical argumentation could be supported by a technological tool, investigating the role of self-regulation while being scaffolded by CSCL scripts and examples. Serrano-Cámara et al. (2014) studied how students become motivated to learn programming concepts in a CSCL setting. Another study investigated how collaborative learning through pair programming can improve pupils’ programming and computational thinking skills (Echeverría, Cobos, and Morales 2019).

1.2. Block-based programming: an overview

There is a growing recognition that computing is an essential skill for all students to master, and consequently block-based programming and computer science courses have entered school curricula in many countries (Mørch, Litherland, and Andersen 2019; Weintrop and Wilensky 2017). Block-based programming relates to visual programming, which became a niche area of programming language research in the mid-1980s, following the invention of graphical user interfaces. Two pioneering visual programming environments were Fabrik (Ingalls et al. 1988) and BLOX (Kopache and Glinert 1988), which made it easier for non-expert and disabled users to learn textual programming (Smalltalk, Pascal, or C) using direct manipulation interfaces (drag-and-drop programme structures analogous to solving a jigsaw puzzle or building with LEGO). Corral, Fronza, and Pahl (2021) presented a recent overview of block-based programming as a tool during the past 10 years. They defined block-based programming as languages and tools that allow

non-professional users to create working software products with little knowledge of the structure and syntax of a regular programming language. One of their arguments is that block-based programming democratises software development by enabling the creation of software products by non-expert programmers, from elementary school pupils who create software products as part of their education to professionals.

With increased access to online programming environments since the millennium, visual programming has reached non-professional communities through hobbies and entertainment (e.g. creating video games and animations). Further innovations in user interfaces for editing code, software tools, and environments that run in the browser are important success factors (Resnick et al. 2009). Another success factor is the availability of an online repository of examples (source code and applications) created by peers. Today, block-based programming environments typically serve as students' introduction to the practice of programming (Weintrop and Wilensky 2017). Scratch is one of the most popular block-based programming languages (Brennan and Resnick 2012; Resnick et al. 2009; Zhang and Nouri 2019). It is known to 'lower the floor' to programming. Other block-based programming languages, such as Alice (Utting et al. 2010), are used by older students for solving problems in, for example, science, technology, engineering, and mathematics (STEM) topics (Zhang and Nouri 2019), and have been used to create specialised block-based languages, such as Robot Blockly for industrial robotics (Weintrop et al. 2017). Older and pioneering block-based environments include Boxer and AgentSheets, for physics education and computer science education, respectively (diSessa and Abelson 1986; Repenning, Ioannidou, and Zola 2000).

1.2.1. Block-based programming in education

Weintrop and Wilensky (2017) emphasise that block-based programming is a useful programming language to use in an educational context, as it has several key features that differentiate it from text-based programming and other visual programming languages. Most notable is the jigsaw-puzzle-piece metaphor, which is used to provide visual cues to the learner about which commands to use. The main advantage of block-based programming is that it reduces pupils' challenges in learning programming language syntax and renders programming more accessible to novices (Sengupta et al. 2013). Other researchers have highlighted micro-worlds and puzzles as metaphors connected with block-based programming and explored the potential of teaching introductory programming from a design

science perspective, focusing on design guidelines and data analysis methods that can be used for designing and improving block-based programming (Pelánek and Effenberger 2022). Namli and Aybek (2022) investigated the effect of block-based programming and unplugged coding on fifth graders' computational thinking skills, self-efficacy, and academic performance. The study included 82 fifth graders in middle school. The authors concluded that block-based programming activities had a moderate effect on computational thinking skills. Jiang et al. (2022) explored the programming trajectories, postulating that block-based programming languages provide effective scaffolding for K–12 students to learn computational thinking. Their main finding was identifying four different roles of block-based programming learners in school: quitters, approachers, solvers, and knowers.

In this study, we used Microsoft's MakeCode for a micro:bit, a block-based programming environment used with a small computer (micro:bit). Combined, Microsoft MakeCode and micro:bit provide an accessible high-level programming environment for embedded devices without sacrificing performance and efficiency (Devine et al. 2019). Programming a micro:bit with MakeCode also allows learners to design and programme physical components, such as servo motors, sensors, actuators, and lamps, which are components of a makerspace. These components can form part of more complex visual artifacts composed of software and hardware.

1.3. Physical computing in education

Physical computing involves the design and composition of interactive objects and enables students to develop concrete, tangible real-world products (Przybylla and Romeike 2014). Accordingly, physical computing can be used in computer science education to provide students with interesting and motivating access to different topics areas of the subject (Przybylla and Romeike 2014). In our case, this is realised by Microsoft's MakeCode (Devine et al. 2019) with a physical micro:bit board connected to different physical components. Sari et al. (2022) studied physical computing in an educational setting, combining hardware such as sensors, LEDs, and servo motors with programming activities, such as problem-solving, algorithm creation, and code writing. They found that STEM-focused physical computing activities developed teacher students' computational thinking skills (Sari et al. 2022).

Smart-thing design is a term used to capture the entire design process of enhancing everyday things, like toys, with computing devices and software capabilities,

emerging from the field of ‘Internet of Things’ (IoT). Smart things are defined as autonomous physical/digital objects augmented with sensing, processing, and network capabilities (Kortuem et al. 2010). These smart things can sense, log, and interpret what is happening with themselves and the world, act on their own, intercommunicate with each other, and exchange information with people (Kortuem et al. 2010). Root, Heuten, and Boll (2019) designed a card-based teaching approach, Maker Cards, using the physical computing device Calliope (an IoT platform) to give children instructions for the hardware and software to help them make their own meaningful artifacts through programming. Gennari et al. (2022) presented a study of a toolkit for smart-thing design with children that included a card-based board game for children, the aim of which is to engage and enable children to design smart things. In the field of interaction design with children, a group of researchers inspired by Papert’s ideas on constructionism have developed materials-based construction kits for children (Eisenberg 2005), supported by design principles (Resnick and Silverman 2005), historical review and analysis (Blikstein 2013), and empirical studies (Sheriff et al. 2017). A study was carried out to promote children’s initiative, positive risk taking, and procedural thinking, all in the context of their bedrooms (Sheriff et al. 2017). The researchers found that interaction with a mechanical construction kit can empower children to develop curiosity about the mechanical world around them, think about risk taking as a potentially positive experience, and think more critically about initiative in the smart home era (Sheriff et al. 2017).

1.4. End-user development and domain-oriented design environments

End-user development (EUD) refers to a set of methods and techniques that allow people who are nonprofessional software developers to create or modify a software artifact (Lieberman et al. 2006), in contrast to the technical development carried out by trained programmers and software engineers (Batalas et al. 2021). It includes, among others, visual programming (Repenning, Ioannidou, and Zola 2000) and domain-oriented design environments (Fischer 1994). EUD researchers have developed tools and environments and tested them in laboratories, organisations, and homes (Fogli, Peroni, and Stefini 2017). We study EUD in educational institutions (schools).

We argue that the usefulness of EUD environments in educational settings can be assessed based on accessibility, flexibility, and purpose. *Accessibility* means the extent to which the EUD environment provides a gentle slope to modification and programming (Mørch and Zhu 2013;

Wulf and Golombek 2001). *Flexibility* refers to the extent to which the tools have a low threshold and high ceiling, allowing many interesting artifacts to be created (Repenning, Ioannidou, and Zola 2000; Resnick et al. 2009; Weintrop et al. 2017). Furthermore, accessibility and flexibility should be measured against *purpose* (e.g. solving technical problems, practicing computational thinking, or learning STEM topics) (Li et al. 2019). Our aim is to understand block-based programming as a tool for explorative learning of STEM topics.

Domain-oriented design environments (DODE) represent a special area of EUD, which we have adopted in our research as a conceptual framework for the future of block-based programming. The three basic components of DODEs are 1) a construction kit with a set of building blocks (design units [DUs]) for creating visual artifacts, 2) an argumentation component that provides access to the design principles of the visual artifacts (referred to as the artifacts’ design rationale), and 3) a critiquing component that analyzes a visual artifact according to a set of knowledge-based rules (Fischer 1994; Fischer et al. 1996). When a design rule is violated, the critiquing component provides feedback to the user with a link to the argumentation (design rationale) for the rule.

DODEs have several applications to education, including:

1. Supporting the creation of an artifact with a construction kit,
2. Signalling potential breakdowns with a critiquing component, and
3. Supporting the exploration of argumentation and design rationale.

Artifacts created with a DODE are externalizations of the designers’ thoughts (Bruner 1996; Fischer 1994). A common denominator of DODEs and block-based programming is component-based design where the user of a computational design environment interacts with the visual components selected from a palette of parts and then combined and modified in a work area (2D or 3D editor). However, most block-based programming environments lack the support of knowledge-based rules and methods and tools for using the rules. These features provide the means for scaffolding and design rationale (argumentation for rules). For example, if we were able to identify structural relations between the components of an educational makerspace (software and hardware), rules would allow for scaffolding. We focus on how empirical researchers can analyze the artifacts with rules, and we provide a scenario showing how the rules can be applied in automated scaffolding.

1.5. Human-centered AI in education

Shneiderman (2020) defined human-centered AI (HCAI) as a promising direction for designing AI systems that support human self-efficacy, promote creativity, clarify responsibility, and facilitate social participation. Furthermore, he identifies three ways HCAI can put humans at the centre of design thinking: 1) acknowledging a two-dimensional HCAI framework with high levels of both human control and automation, 2) a shift to empowering people away from descriptions of intelligent autonomous systems and toward explanations of powerful tool-like applications, 3) a governance structure that describes how software engineering teams can develop more reliable AI systems. Fischer (2021) suggests that EUD and AI should be integrated, and he identifies several areas of HCAI that intersect with the aims of EUD (empowering end users rather than replacing them), including IA, explainable AI (XAI), democratising AI, ethics and trust, common ground, and shared understanding. The latter two areas overlap with research topics in CSCL (Stahl 2016).

Yang et al. (2021) argued that AI can evolve into HCAI by developing AI from a human perspective considering human conditions and contexts, particularly focusing on how AI technology can enable different forms of human performance. They explored how AI can be used to evaluate new design methods, tools that can advance AI research, education policy, and practice, with an overarching aim to leverage AI's potential to educate, train, and improve the performance of humans rather than doing interesting things for them. According to Yang et al. (2021), the shifting of AI research toward creating HCAI designs that consider human conditions and have a human-oriented approach should be seen to augment (rather than replace) human intelligence with machine intelligence.

HCAI in education can be approached, for example, with an AI-enabled conversational robot (chatbot), AI-enabled personalisation, smart content provisioning, learning pathway guidance, learning design support for teachers, course recommendation for students, intelligent assessment, evaluation of course essays (Mørch et al. 2017), automatic question generation, and AI-enabled plagiarism detection (Yang et al. 2021). In this article, we explore the use of HCAI in the context of 1) visual artifact analysis and 2) automated scaffolding by an AI-enabled conversational robot (chatbot) in a physical computing STEM context.

Although AI in education is often touted as an emerging field in educational technology, many educators are uncertain about how to take full pedagogical advantage of the technology in the classroom. Akinwalere and

Ivanov (2022) presented examples of introducing AI in higher education, discussing its possibilities and risks. Our empirical findings address challenges and opportunities for K–12 education.

1.6. Rationale and research questions

In block-based programming, syntax is validated through the automated connection of blocks but there are no built-in mechanisms that support the user in combining blocks to reach personally meaningful or educational goals. Our approach to HCAI involves two ways of applying knowledge-based rules: by educational researchers in visual artifact analysis and for designing scenarios in which pupils interact with automated scaffolding in block-based programming.

Accordingly, we ask the following research questions:

How can empirical researchers take advantage of HCAI in education by using knowledge-based rules in block-based programming environments to 1) analyze CSCL artifacts using visual artifact analysis (RQ1) and 2) propose scenarios of computer-based (automated) scaffolding (RQ2)?

In sum, our related work draws on different aspects that make out the research questions. We focus on block-based programming in a collaborative context in which pupils use a physical micro:bit for solving subject specific tasks in class. Following this, we explore how this can be taken one step further by applying automated scaffolding or HCAI to offload teacher's workload described in a scenario. We address these questions in this article, which is organised as follows. In section 2, we present our research methods. This includes a description of the pupils and their selection, the educational task and setting, the data collection techniques, and the analytical framework for analyzing the data. We illustrate the materials (software and hardware) used in the classroom. In section 3, we present the study results, including a method for empirically analyzing visual artifacts using rules and for proposing computer-based scaffolding scenarios. Finally, we discuss our results, answering the RQs and comparing our findings with those in related works.

2. Materials and methods

2.1. Analytical framework

We analyzed the empirical data on pupils' group interactions in computer-mediated settings in science education from two perspectives: 1) research analysis of collaborative learning processes and artifacts and 2) requirement analysis to identify opportunities for

computer-based scaffolding using HCAI. We created an analytical framework to facilitate this endeavour, particularly to understand the intertwining of discursive (group interaction) and visual (programming, making) activities. The concepts in the analytical framework are derived from central topics in CSCL and EUD. We used these concepts in an integrated effort as thematic codes to make sense of the empirical data in our analysis. The analytical framework builds on previous work (Andersen, Mørch, and Litherland 2021) but is extended by detailing various rules in the visual artifacts. Our aim is to understand the collaborative learning of knowledge and skills in specific STEM domains (e.g. math, biology, and physics) when block-based programming is used as a method. From the CSCL perspective, we highlight the collaborative learning process and the common artifacts that are created (Stahl, Koschmann, and Suthers 2006). Additionally, from the perspective of EUD, we draw on the concept of DODEs (Fischer 1994), which allows us to define knowledge-based rules as relations between design units (DUs). Combining these concepts from two different research traditions, computer science and social science, provides a group interaction perspective on domain-oriented design environments, which, to the best of our knowledge, is novel. The analytical framework is presented in Table 1.

Personal perspective is the understanding an individual contributes during group work. Information sharing occurs when the participants share their personal perspectives. It is a central element of collaborative learning, as it starts all other meaning-making processes (Stahl 2006). Negotiation occurs when the group begins to establish a shared perspective through which they construct and maintain a common understanding of the task and its solution. Individual group members must negotiate multiple personal perspectives to create one that can be shared and to affirm that the meaning is, in fact, shared (Stahl 2016). According to Stahl

(2016), group cognition is a goal of collaborative learning and occurs when multiple people participate in coherent interactions leading to cognitive accomplishments that are best analyzed, at least in part, at the group level, rather than attributing contributions and agency entirely to individual minds. Scaffolding is a metaphor in the collaborative learning context, describing how teachers and experienced peers offer temporary support structures to assist learners in developing new understandings, concepts, and abilities by providing feedback and support (Hammond and Gibbons 2005).

DODEs are computational environments whose value is not restricted to the design of software artifacts (Fischer 1994). Based on DODEs, we adopted the concept of DUs, which are the basic objects in the design environment (i.e. the smallest units that designers manipulate during a design process). DUs can be specific domain objects referring to elements of an assignment given by a teacher as well as objects that are more technical or general in nature. In this context, they are the hardware (microcontrollers and physical things) and software objects (code blocks) users interact with when they create things and programme them in a makerspace. We refer to DU configurations as visual artifacts or technical objects (Andersen, Mørch, and Litherland 2021). They are distinguished from discursive (CSCL) objects, which are composed of oral or textual utterances that contribute to shared understanding. Rules are relations between two or more DUs and define the knowledge embedded in a visual artifact or technical object. These rules can be identified through visual artifact analysis (section 3.1). Examples are previous solutions (software and hardware) that can be reused in new designs. We broadly interpret the concept of examples to include aspects of testing and refining a first version of something that is almost finished.

2.2. Research design and methods

In the following section, we present the research design and specifics of the data collection and analysis. As the intervention described below was executed during the COVID-19 pandemic, all classes were held remotely using a video conferencing platform (Zoom).

The project employed a design-based research (DBR) approach, which involves collaboration between different stakeholders in the design and implementation of educational interventions, seeking to contribute to the development of both theory and practice (McKenney and Reeves 2018). Hence, the participating teachers and pupils did not act as ‘informants’ per se but as partners in the design and execution of the project, as detailed below. As opposed to lab-based approaches,

Table 1. Analytical framework: A set of concepts derived from CSCL and DODEs.

Analytic concepts derived from CSCL	Analytic concepts derived from DODEs
1. Personal perspective	1. DU (separate building blocks to be combined with other DUs to form more complex designs)
2. Information sharing (from me to you)	2. Rule (relation between two or more DUs).
3. Negotiation (you and I decide what to focus on)	3. Example (complete design for reuse)
4. Group cognition (shared meaning/perspective)	
5. Scaffolding (help from teacher/senior peer)	

where pedagogical theories are tested in isolation from a general context, DBR interventions are executed ‘in context’ through iterative development (Hoadley 2002). The data used in this paper were collected over the course of an academic year.

Research Design. The DBR intervention comprised a series of 16 three-hour science classes developed with the aim of supporting gifted pupils in learning accelerated science content using block-based programming as part of a makerspace. The students were randomly divided into six groups, and a teacher was assigned to each group. Each group completed all 16 classes. Teachers in upper secondary schools taught the classes while the students were still enrolled in lower secondary schools. These more qualified teachers were required to support the goal of accelerated learning, as the intervention covered learning goals from upper secondary schools. The teachers started each online class with traditional introductions to the scientific topic of the day, followed by introductions to any relevant technical materials (physical or digital) necessary for individual or group tasks. This included showing the students examples of MakeCode scripts they might apply. The teachers divided the class into groups (breakout rooms in Zoom) comprising three to four pupils, where they worked on practical assignments employing programming skills and content knowledge just introduced by the teacher. The pupils were free to solve the tasks creatively in any reasonable way they chose. As the students worked remotely, they were encouraged by the teachers and researchers to share their screens, communicate using voice and text chat, and point to their project with their web cameras. The concept of design in DBR (design as intervention) is different from that in DODEs (design as creation). Thus, we use the concept of design from the perspectives of both behavioural science and information technology (computer science).

Participants in the Study. The participants in the research project are gifted children who were selected based on several checklists indicating their interest and ability in addition to meetings with the pedagogical-psychological service in the municipality. The participants were ability tested by certified psychologists with WISC-V (95% IQ > 120) to ensure that they were defined as gifted. Mönks and Katzko (2005) defined giftedness as the individual potential for exceptional or outstanding achievements in one or more domains, which reflects the concern for practical, educational issues in a developmental context. Renzulli (1978), one of the pioneers and most well-known researchers in the field, defined gifted children based on a three-ring model of gifted behaviours consisting of three types of human traits that are dynamically dependent on each other:

1) task engagement, 2) task persistence, and 3) creativity. It should be emphasised that defining giftedness in children requires multiple qualities and not just a high IQ score. These key qualities include motivation, high self-esteem, and creativity (Reis and Renzulli 2004).

Building on Renzulli (1978), Mönks (1992) proposed the ‘multifactor model,’ adding an education dimension by focusing on school, peers, and friends as factors to be considered for understanding gifted behaviour in a school context. Later research suggested two different educational approaches to adaptive education for gifted pupils (Freeman 1999): acceleration and enrichment. Acceleration means increasing the tempo of the learning process by moving pupils to a higher level, an older age group, or compacting the material they must learn. Enrichment involves going into more depth about a specific topic to be learned. Our intervention involved aspects of acceleration and enrichment. The competence goals of the learning design were 3–5 years above the pupils’ academic year, and we focused on creating exploratory and open-ended tasks where the pupils had many alternatives for task completion and could work together.

A core principle of gifted education is individualisation and differentiation (Mönks and Katzko 2005). Therefore, we wanted to create learning designs for a much-needed learning programme in which gifted children would be motivated to participate. Our unit of analysis is collaborative learning (group interaction) and exploratory learning using programming for adaptive learning in a STEM classroom. Furthermore, programming has become a central topic that educational policymakers argue all students should learn (Bocconi, Chiocciariello, and Earp 2018). Current research explores how programming can be integrated into school subjects and the implications of doing so. Gifted children provide a good starting point for an explorative study of programming in STEM since they are motivated to participate and need adaptive education. We studied the visible emerging social practices that we could capture with video recording of the pupils when they were collaborating, implying that the background of the children is not the focus of the study.

Data Collection. The collection of empirical data in this study was a complicated process that was only possible through collaboration with the participating students and teachers. Five researchers and one assistant participated in the data collection. Students, teachers, and one or more researchers and/or assistants were all synchronously present in the three-hour online Zoom classes, which were recorded by the researchers. No data in this paper involved physical co-presence; all classes were online. Except for classes where the

students completed pre- and post-tests (not included in the scope of this article), parts of all classes were recorded, focusing on active group work sessions. For these recordings to successfully capture a maker space on a computer screen, the students and teachers had to digitally share their materials with the researchers and the class, using webcam and voice/text chat. We see this as a conceptual shift from research *on* students (and teachers) to research *with* students (and teachers). The students and teachers did not record themselves but were essential in making decisions about what and when they wanted to share content in the recordings. The researchers were present during the online classes and sometimes proposed that the students shared materials or explained their thinking verbally. This data collection approach is referred to as participant observation, where the observers are sometimes also participants (Jordan and Henderson 1995). Using this method, we collected approximately 70 h of screen recordings. Prior to the intervention, all participants' parents/guardians signed a written consent form with information about the project, including data collection and storage. All collected data were stored using the highest level of security offered for sensitive research data. Additionally, all names used in this paper were pseudonyms. The study was approved by the Norwegian Centre for Research Data.

Data Analysis. The data were analyzed using a two-step approach, combining thematic analysis (Braun and Clarke 2012) and interaction analysis (Jordan and Henderson 1995). We looked for instances of interaction informed by our analytical framework (top-down) as well as for emergent themes from the data material itself, known as an abductive approach (Reichert 2014). We emphasised both the participants' 'voices' as inductive and applied our analytical framework, which is a top-down, deductive approach. The group was our main unit of analysis. During the deductive (top-down) part of the analysis, we explored the material and coded instances based on the framework. The inductive (bottom-up) part of the analysis allowed for the emergence of codes outside of the established framework, such as block-based programming, collaboration, and programming integrated with the subject material. Using the interaction analysis (Jordan and Henderson 1995), we found selected episodes of data to serve as examples of our findings and divided them to analyze them in detail, focusing on the sequential interaction and development of understanding or discursive objects. In practice, deductive and inductive coding was performed individually and separately by the authors. Before we came together and analyzed extracts, each researcher had made a tentative selection

based on our research questions. During these data workshops, we focused on the development and application of the analytical framework and on any emergent codes. The final selection of examples to include in the paper was made by the first author. We show two examples in section 3.2. Interaction analysis allows the researcher to focus on the bottom-up perspective of human interaction (talk, action) with other humans, space, and objects/artifacts in the immediate environment (Jordan and Henderson 1995).

2.3. Materials and activities

We provided the pupils with a toolkit including the different physical and digital materials they needed to solve the tasks assigned by the teachers, and we refer to the materials as DUs (Figure 1).

The data used in the study involved the following subjects: mathematics, biology, and physics. The pupils were given tasks to learn one or more domain-specific concepts using a micro:bit (the physical controller) and creating a block-based code connected to it to solve domain-specific tasks assigned by the teacher. MakeCode is a web-based environment that facilitates block-based programming for writing the code that controls the micro:bit (MakeCode 2021). One task example is to make the micro:bit function as dice (Figure 2; Andersen, Mørch, and Litherland 2021), another is to make the micro:bit into a burglar alarm, which we will study more in depth in section 3.

Methodological Limitations. Generalizability in qualitative research includes two components: purposive sampling and theoretical sampling. In purposive sampling, a case is chosen because it illustrates the feature or process of interest. In theoretical sampling, a case is chosen based on a theory or perspective (Silverman 2005). In this article, purposive sampling was used since we selected a case study that focused on how processes of collaborative learning emerge when gifted pupils work with block-based programming. It should also be noted that the participants are gifted. Hence, they have different special education needs and intellectual backgrounds than non-gifted pupils in most classrooms, which may affect the generalizability of the findings. We made this choice because the unit of analysis in the study is the processes emerging between the gifted pupils when using programming for solving subject specific tasks in class. This implies that the individual pupils are in the background, representing the context of the interventions, but are not the unit of analysis. The analysis focuses on representative CSCL artifacts collaboratively created by the pupils in group work.

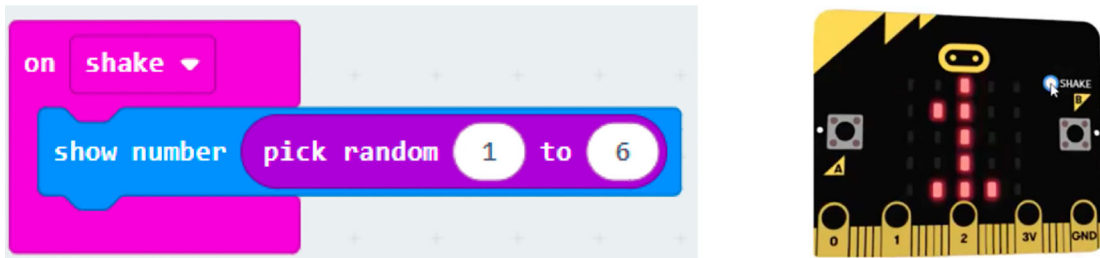


Figure 2. Visual artifact for simulating throwing dice by calling a random number function and displaying its values on a micro:bit display. The visual artifact comprises four DUs, three software code blocks (left), and one hardware unit (micro:bit, right). The relations between the DUs define three types of rules.

- *Latent rules* are tacit (taken for granted, automated) and general (applicable across multiple domains and tasks). For example, composition and sequencing.
- *Generic rules* are distinguished from latent rules by being explicit (learners must be consciously aware when choosing them) and feature generality. Most programme constructs.
- *Domain-specific rules* are distinguished from the two other types by being explicit and pertaining to a specific domain or task. Specific functions, variable names, and values.

3.2. Application of rules in empirical research

In this section, we present empirical data that illustrate how the pupils we studied used the block-based code to experiment and explore subject area topics in STEM education. Each section is divided into four subsections: 1) contextualising the extract, 2) raw data (informants' voices), 3) technology object (software and hardware being composed), and 4) discursive object (a sequence of utterances advancing the groups' common understandings). The transcript notations we used include the following symbols: ((text)): participants' actions, [text]: researchers' clarifying comments, (.): long pause in interaction, and ...: pause in interaction. In all

extracts, one of the pupils shared their screen so that the other pupils could see what they were working on.

3.2.1. Example 1: programming the micro:bit to function as a burglar alarm

Contextualising the Extract: The data extract in Table 2 is derived from a class in which the pupils made the micro:bit work as a burglar alarm by connecting different physical and digital artifacts. The subject-specific learning goals were derived from physics, covering topics like electricity and conductivity. The pupils used materials from the provided toolkit, such as tape and aluminum foil, to create the micro:bit alarm. The texts in parentheses are objects the pupils refer to or act upon. William and Damien are working remotely on their separate solutions to the burglar alarm in the zoom breakout room. William is filming his project with his web camera and talking about the issues he has connecting the different physical parts. His classmate Damien is watching and commenting. The teacher joins the breakout room mid-extract.

Technology Object. The development of the technology object is partly shown in Figure 4 (two snapshots), which is William's version of the part of the device that has three wired components—a micro:bit, a servo motor, and a camera set inside a wooden frame



Figure 3. Visual (digital) artifact (left) with a micro:bit (right) for controlling audio in a simple burglar alarm.

Table 2. Relations between physical artifacts.

Turn	Participant	Utterance	Analytical concept
1.1	William	Okay, new problem. The servo holds, but it's lifting the whole camera. How should I connect the camera [to the burglar alarm]?	personal perspective, design unit (2), latent rule
1.2	Damien	[70 s later] But the servo should be taped on top of the camera.	negotiation, design unit (2), scaffolding, latent rule
1.3	William	On top of the camera? Yes, but no, it has to be next to it. I thought about it, but this is the easiest way if I just find a way to attach the camera.	design unit (2), personal perspective, generic rule
1.4	Damien	But you don't have any support for the camera on the one side, on your left side of the camera.	negotiation
1.5	Teacher	Can you take a piece of tape? I'm thinking that if you put a piece of tape—You know you taped the other side [of the camera], but if you put a piece of tape across the whole camera.	scaffolding, design unit, generic rule
1.6	Damien	Yes, that's what I'm thinking too.	shared perspective (2)
1.7	William	Across the whole camera?	design unit
1.8	Damien	Start on top of the camera, the part facing the ceiling, and then you stick the tape from there to the underside of your 'boat' [the foundation on which he is building the burglar alarm].	scaffolding
1.9	William	Yeah, so across the whole screen at the top.	shared perspective (2)
1.10	Damien	Yes.	
1.11	Teacher	Yes.	
1.12	William	I hadn't thought of that. Good if it works. ((tapes))	group cognition (3)

resembling a boat. The problem that led to this physical configuration is an earlier configuration in which the motor's movement pushed the camera such that it was lifted and stopped working. In turn 1.1, the relation between two of the DUs is established (camera and servo), which we interpret as a latent rule, meaning that the connection is a result of trial and error by William rather than deliberation. Damien suggests another, perhaps brute force, idea to connect the two devices by taping them together. This should be considered an application of a generic rule, as the act is justified through several rounds of deliberation. William presents a generic rule between these two DUs when he says that the camera needs to stand next to the burglar alarm. In turn 1.5, the teacher tries to help William by building on Damien's previous suggestion (latent rule) to use tape, which, by now, is a generic rule for how to connect the camera to the burglar alarm.

Discursive Object. Referring to the conversation unfolding in Table 2, Damien suggests an idea for solving

the problem with the servo motor and the camera, which starts a collaborative inquiry process in which the group begins to establish a shared perspective (Stahl 2016) by William's proposal of a personal perspective (a problem). Damien suggests his idea for improvement to help and scaffold William. Considering Damien's idea, William (in turn 1.3) responds by again sharing his personal perspective. Next (turn 1.4), Damien challenges William's statement, reflecting a negotiation process. Abruptly, in turn 1.5 the teacher intervenes and provides a scaffold by elaborating Damien's idea about taping the whole camera. Damien provides additional elaboration, and they reach a tentative shared perspective (Stahl 2016). Damien (turn 1.8) helps William by clarifying how the idea the teachers suggested can be accomplished and, thereby, provides a new scaffold. Thus, they affirm their shared perspective in turn 1.9–1.11, which leads to an agreement on how to solve it, referred to as group cognition (Stahl 2016).

Our data indicate that scaffolding is frequent on different levels (latent, generic, and domain-specific) by different actors (peer, teacher) to help the pupils move forward and further their understanding of scientific concepts. The next extract also shows examples of peer and teacher scaffolding at the domain level and a scenario of computer-based (automated) scaffolding.

3.2.2. Example 2: the burglar alarm is not working

Contextualising the Extract. This example is derived from the same context as example 1 (programming a burglar alarm) and occurred sometime after the first example. Immediately prior to the extract in Table 3, William shared his screen and explained the code and how he created the burglar alarm by modifying the code and discussing how to make the burglar alarm sense movement by changing the axis parameter of the accelerometer. The two pupils test and discuss how the burglar alarm reacts to motion when the door is opened, focusing on the acceleration value of the motion sensor. They find that the motion detector is not sensitive enough, and thus they discuss how to change the code to make their device more sensitive to the pupil's door movement.

Technology Object. Figure 5 shows the MakeCode a pupil created to control the burglar alarm by connecting three different micro:bits to detect a door opening. The code reflects the complexity of the programming process in the number of relations established between the DUs, which are numerous and can be gleaned from Figure 5 (and partly from Figure 4). The code connects the micro:bits and the GoPro camera through wires, radio signals, and the servomotor. The micro:bits built in the accelerometer detect door motion, which is measured

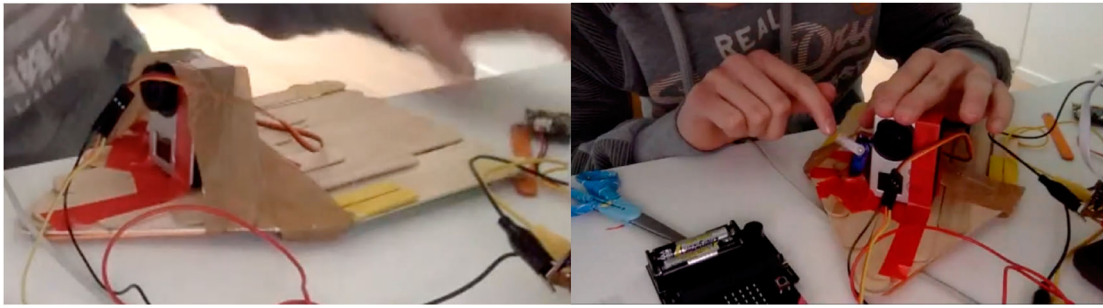


Figure 4. Two screenshots of the burglar alarm created by William, where the camera is taped (left); to the right we see how William tests if the camera is attached to the burglar alarm when the servo motor starts (points with finger).

by acceleration in milli-g (thousandths of the standard gravity), and its value is added to a conditional block in MakeCode. The pupil selects this code block to trigger an alarm sound by connecting it with the ‘play tone’ block that is part of its subassembly. The example brings latent, generic, and domain rules to the foreground. For example, the pupil uses problem-solving as an implicit rule that is part of the taken-for-granted process of solving science tasks in a classroom setting. Latent rules are also evident in the composition and sequencing of code blocks. This can be seen in the multiple subassemblies that are in the process of being included or excluded in the overall execution and seen as alternative configurations outside the main execution block in Figure 5 (see the unused event handler blocks ‘on button pressed’ in the left image of Figure 5, which simply displays a count). Furthermore, generic rules define the different wiring of physical components (Figure 4) and the choice of logic for the programme execution using conditionals and set operations (Figure 5). A domain rule is associated with using the *acceleration* conditional (if) block, which is also a key concept in Newtonian mechanics and part of the physics topic taught (which also includes electricity). When the pupils are confronted with programming terms that also relate to domain-specific concepts, they can connect theoretical concepts with practical activity, which is one of the aims of educational makerspaces.

Discursive Object. The extract started out with William (turn 2.1) stating that the micro:bit does not work; it only made a high beeping sound. This problem triggered an understanding process, which eventually led to a shared perspective on the task and group cognition, where all members shared the same perspective (Stahl 2006) about acceleration as a physics topic (originating from the teacher’s perspective) as well as about developing devices that work (originating from the pupils’ perspectives). Consequently, Damien (turn 2.2) asked what cut-off point the acceleration condition was set to, after which a series of tests led them to a shared understanding of the acceleration concept. This

understanding was sufficient for the programming process, an iterative process of making and building understanding that led to group cognition (Stahl 2006) in turn 2.17. The discussion of the milli-g, which is the unit of measure for the accelerator, reflects the domain-specific knowledge connected to the physics task they were working on. It is interesting that they connected the learning of domain knowledge to the programming process. This might be a strategy they employ when they meet challenges and lack the necessary scaffolding. The pupils discussed how to select the cut-off point for the accelerometer condition to make the burglar alarm work properly, and during this process, they shared information and negotiated their ideas, some of which were related to physics and others to collaboration and problem-solving.

Scaffolding the Discourse. Damien’s response in turn 2.6 revealed that he did not know about the concept of axes required for using the acceleration function in MakeCode. Consequently, William explained how to determine what axis he was using, thus peer scaffolding (Hammond and Gibbons 2005). However, Damien (turn 2.8) did not understand it, so he again asked how to do this. This reveals that the pupils lacked the background knowledge (generic or latent rules) of how the accelerometer in the micro:bit works, which could potentially connect with Newtonian mechanics but could not be achieved by peer scaffolding alone. The teacher (turn 2.9) intervened and asked Damien, who had a working burglar alarm, if he used the z-axis. This is another example of a scaffold (Hammond and Gibbons 2005), in this case from the teacher, drawing on Damien’s (lack of) pre-understanding. In turn 2.10, Damien again bailed out and switched the activity to a more hands-on one (attaching the micro:bit to a door), which turned out to be a game changer for lifting the conversation to a higher (more theoretical) level. In turn 2.11, a fictitious agent Jack addressed the issue (Damien’s lack of knowledge of the z-axis) by explaining through a simulated voice what an accelerometer is and

Table 3. Relations between physical and digital artifacts.

Turn	Person	Utterance	Analytical concept
2.1	William	This is bad, it [the micro:bit] is screaming constantly.	design unit
2.2	Damien	What acceleration do you have, William?	information sharing
2.3	William	It's 200 milli-g, so it might be too low?	personal perspective, negotiation
2.4	Damien	Mine is at 500 milli-g.	personal perspective
2.5	William	Which axis are you using?	design unit (small DU)
2.6	Damien	Eh, how am I supposed to know!	
2.7	William	It tells you! In the same place [block], it's easy to see. It says x, y, or z, and then power, or—what's it called again? Strength [force, see Fig. 3]	generic rule, domain-specific rule, personal perspective
2.8	Damien	Eh, let me check. It says ... x. What should it be?	design unit
2.9	Teacher	Didn't you use z, Damien?	design unit (z); latent rule
2.10	Damien	I think z is best, but how did you attach the micro:bit to the door?	design units; generic rule
2.11	Simulated Agent Jack (automated scaffolding)	The micro:bit is equipped with an accelerometer, which measures movement along three axes: x (movement from left to right), y (movement forward or backward), and z (movement up and down). There is a variable for each axis, which returns a positive or negative number that indicates the milli-g forces. When the accelerometer variable's value is 0, the forces pulling each way on the axis are equal. In the last minute or so, I've mainly registered movement on the z-axis.	design unit (3), generic rule, domain rule, scaffolding
2.12	Damien	Yes, I just taped mine [the micro:bit] to the door like this. ((shows with camera?))	information sharing, personal perspective, design units (2)
2.13	William	But have you taped the micro:bit to it [the door]? It [the alarm] didn't work when I tried.	negotiation; design units (3)
2.14	Damien	Yes, it worked. I taped it like this and at a low-turning radius and put it close to the hinges in a way. Also, I set it to 'y' and just made it a 1 so that it registers the smallest of movements. It's taped	information sharing, negotiation, design units (4), domain-specific rule

(Continued)

Table 3. Continued.

Turn	Person	Utterance	Analytical concept
		firmly so it doesn't move unless someone makes it move.	
2.15	William	Yeah, that's good. Okay, so it's not beeping yet. ((has taped the micro:bit to the door and tests the design by opening the door))	information sharing, negotiation, design units (4), domain rule
2.16	Damien	Is it beeping? ((hears sound from micro:bit)) Yes!	negotiation, design unit
2.17	William	Yes, it's working. But 500 mg was a little too much maybe.	shared perspective, example, negotiation
2.18	Damien	Yes. In my experience, it was difficult [for the micro:bit to detect movement] if I opened the door smoothly, but if the door suddenly stopped moving or as soon as I closed it again, it registered the signals quite fast.	example, domain rule, design unit (2), shared perspective
2.19	Teacher	It makes sense since it's acceleration. If you open it [door] slowly and steadily, it doesn't change. Then you have zero acceleration.	domain rule, design unit, scaffolding
2.20	Damien	Sure ...	group cognition

how it has built in the z-, y, and x-axes. The teacher intervened the second time in turn 2.19 to orient the higher-level discussion toward domain knowledge by explicitly invoking the concept of acceleration: 'It makes sense since it's *acceleration*. If you open it [the door] slowly and steadily, it doesn't change. Then you have *zero acceleration*' [emphasis added]. This extract is interesting in how the pupils used programming as a method for learning Newtonian physics, and in doing so, revealed the underlying implicit and explicit rules that gradually became more specific as the students' orientations aligned with the teacher's orientations through scaffolding.

4. Discussion

In this section, we address RQ1 (analyze CSCL artifacts using visual artifact analysis) by comparing our findings with those obtained in the related work we surveyed in section 1. RQ2 is addressed in Section 5.

4.1. Analyzing CSCL artifacts using visual artifact analysis

Our interest in the visual artifact analysis of CSCL artifacts is based on 1) prior work in domain-oriented

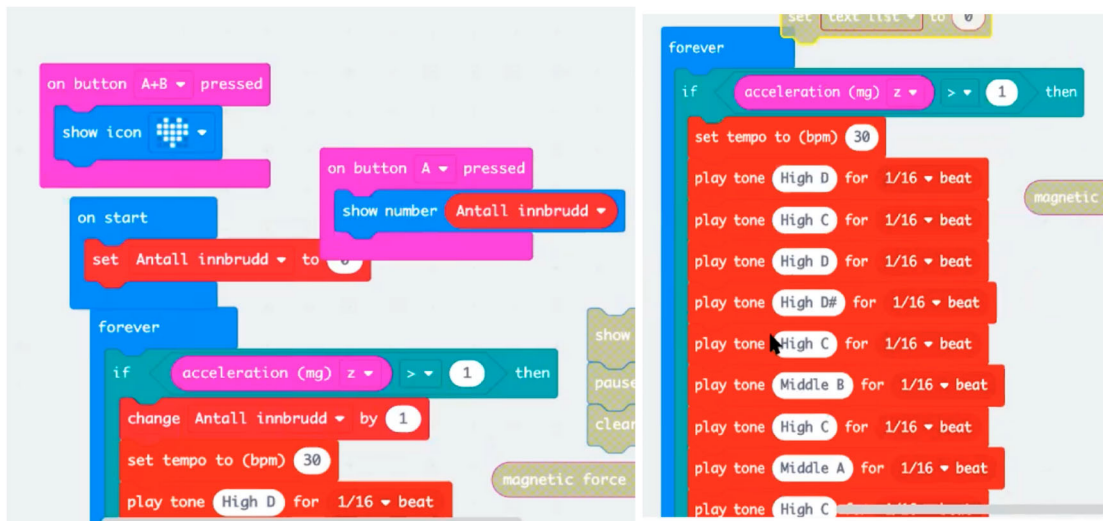


Figure 5. Parts of the code developed by one of the students to interact with the micro:bit-controlled door alarm.

design environments and end-user development (Fischer et al. 1996) and 2) identifying a technique to supplement verbal interaction analysis in CSCL research (Jordan and Henderson 1995). Visual artifacts in DODEs were kitchen designs composed of DUs (appliances and cabinets) and rules for how the DUs should be combined according to the kitchen design rules, thus embedding domain knowledge in visual artifacts. In this study, we proceeded by reverse engineering the tangible results of a learning activity, whereby rules were gleaned by decomposing visual (STEM) artifacts into DUs and relations. Latent, general, and domain rules were identified. Latent rules emerge as implicit rules based on the tacit knowledge associated with the participants' experiences and prior knowledge (i.e. learned in and out of school) and automated behaviour in programming and making. Generic rules relate to the conscious actions performed by pupils when they are developing things and programming by composing DUs. Domain rules are more specific and oriented toward a knowledge domain, task, or learning goal. We have focused on one domain concept as an example, *acceleration*, which the pupils chose as a control mechanism to detect door motion so that the micro:bit could function as a burglar alarm, triggered by accelerated motion. This is one of the many possible solutions to the task, and the two pupils we followed chose this solution.

4.2. Visual artifact analysis extends and complements interaction analysis

What we learned from our visual artifact analysis that we could not obtain from the interaction analysis is

associated with two trajectories of development: a technology object and a discursive object. The former depends primarily on VAA, while the latter depends on interaction analysis. We consider both important for understanding the pupils' collaborative learning, as two different CSCL artifacts that complement and refer to each other during the learning process. On one hand, our visual artifacts are mediating artifacts, such as the artifacts used in interaction analysis (e.g. deictic references). On the other hand, visual artifacts are dynamic entities; they are objects that evolve with their own logic (from a simple to a more complex visual artifact, involving various rules that define relations between the DUs as they form part of subassemblies) (Fischer et al. 1996). This is different from the logic of developing understanding in verbal discourse (from personal to shared perspectives) (Stahl 2006). This implies that visual artifacts are an evolving context for verbal discourse; thus, the two objects (technology vs. discourse) define separate but interdependent trajectories that evolve in parallel.

5. Directions for further work

In this section we discuss RQ2 (propose scenarios of computer-based (automated) scaffolding).

The underlying rationale for suggesting automated scaffolding prompts with the micro:bit is that we believe IA and HCAI can be useful for teachers to offload their work, implying that automation will not lead to AI (replacement). In a programming class, a teacher is supposed to help up to 25 pupils simultaneously, which can be a daunting task or, at best, an exercise in prioritisation. We focused on simulating what an intelligent

assistant can provide in terms of scaffolding by giving examples of useful information. The technical implementation is outside the scope of our work as educational researchers and is a direction for further work.

It would be quite helpful to the teachers if the micro:bit could have built-in sensors that provide immediate feedback to the pupils during their programming process in the future. Affordances built into the technology can indicate actions for domain expert end users to perform useful tasks. An illustrative example is when playing with a train construction kit as a child—where each train had different magnets connected to their ends. If you combined the cars with the right magnets (positive and negative), they would ‘click’ and connect, resulting in a long train of carriages. If you tried to connect two carriages that were not compatible, the objects would signal this by their magnets’ opposing forces. The magnets can be seen as a built-in affordance that provides information about how to connect the train carriages. Moreover, they provide an entry point to the physics of electromagnetism, where a skilled teacher can align students’ trajectories of hands-on making and building understanding into a new path of focused understanding enabled by scaffolding.

We sought to identify how an intelligent agent, Jack, can provide automated assistance to the learners based on detecting ‘broken’ rules. Some situations in which HCAI can help the teacher include a pupil trying to connect two parts that are not compatible (e.g. the micro:bit could change its colour to red). Another example is providing scaffolding for using radio signals in the micro:bit to connect two microcontrollers. One of the controllers could signal a working line by blinking a green light. Our hypothesis is that by making the physical components more intelligent through built-in affordances and automated feedback, they could function as scaffolds and make it easier for the teacher to help the students who are struggling and need elaborated help. In addition, many teachers often lack detailed programming knowledge. The development of AI support systems requires careful consideration of the task requirements and finding the right balance of teacher assistance and automated help.

6. Summary and conclusions

We addressed how can empirical researchers take advantage of knowledge-based rules in block-based programming environments to analyze CSCL artifacts using visual artifact analysis (RQ1) and proposed scenarios of computer-based (automated) scaffolding (RQ2) by analyzing two visual artifacts in terms of extracting rules (section 3.1) and presenting and analyzing the

development of more complex visual artifacts that explored the application of the rules (section 3.2). In summary, we report the following findings.

- Three types of rules were identified between the DUs with VA: latent, generic, and domain-specific rules,
- Two types of CSCL artifacts (technology and discussions) intertwine and develop in parallel,
- A scenario of computer-based scaffolding that offloads domain-specific scaffolding from humans to computers.

We identified three types of rules using visual artifact analysis. Three types of rules were identified and named: latent (tacit everyday understanding), generic (explicit everyday understanding), and domain-specific (explicit scientific concepts). We used the rules as analytical concepts in the empirical analysis of two verbal data extracts (examples 1–2) and in the development of discursive objects (represented in verbal transcripts). The occurrences of rules were identified in the direct or indirect mentions of relations between DUs by the participants. Links to the corresponding visual artifacts are shown as image snapshots in the figures. A more comprehensive series of successive snapshots would reveal a more detailed process by depicting the increased complexity of adding DUs and subassemblies during the making process, allowing for finer-grained VA and better contextualisation of the discursive object. This is part of future work, which also includes testing whether the rules are also relevant in another domain.

With the combination of VA and interaction analysis, two types of CSCL artifacts emerged: technology and discursive objects. The snapshots that comprise the evolution of the technology object were more specifically related to the visual, technical, and physical characteristics of the micro:bit and how to connect it with related components. In contrast, the discussions connected to the discursive objects were more general and focused on creating a shared understanding of how to solve the task, collaborate with peers, solve technical problems, and involve domain (physics) knowledge by using scientific concepts (e.g. acceleration). The latter was, to a larger extent, dependent on the teacher’s (or more capable peers) scaffolding to increase task completion accuracy.

We suggested a seamless transition to computer-based (automated) scaffolding in the form of simulating a teacher’s behaviour as a facilitator in a makerspace. Automated scaffolding in a classroom setting has great potential for helping teachers when many pupils ask for help of different types, for example, to augment human decision-making and provide timely feedback.

However, there are also negative consequences. One issue is that makerspaces and knowledge-based rules, to a large extent, are contradictory terms (makerspaces are problem-solving spaces for technology enthusiasts and scaffolded bottom-up by peers and not imposed top-down by rules). Second, teachers may have little time in a classroom with 25 pupils, although 25 micro-bits may come handy. Additional help will be needed to resolve technical difficulties because teachers often do not have detailed programming knowledge.

Regarding limitations, our study included participants who were drawn from a pool of gifted students. In educational research, it often is assumed that generalisations cannot be drawn from a small case study or small sample of case studies (Silverman 2005). If this study had been conducted in a classroom with non-gifted pupils, the outcome might have been different. However, our exploratory study allowed us to develop new hypotheses and a new research method for educational makerspaces. As such, the findings may have implications for how teachers plan and design future teaching lessons when using block-based programming integrated in a subject. It also provides an example of how HCAI can profit from such learning designs.

Acknowledgments

This research is part of the research project ‘Programming in school’ (ProSkap) funded by Oslo Regional Research Fund in Norway. Thank you to Ellen Egeland Flø for helping in organisation of the educational activities presented here.


Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Regional Research Fund of The Research Council of Norway.

ORCID

Renate Andersen  <http://orcid.org/0000-0002-1206-2140>
Anders I. Mørch  <http://orcid.org/0000-0002-1470-5234>

References

- Akinwalere, S. N., and V. Ivanov. 2022. “Artificial Intelligence in Higher Education: Challenges and Opportunities.” *Border Crossing* 12 (1): 1–15.
- Andersen, R. 2019. Mutual Development in Online Collaborative Processes. Three case studies of artifact co-creation at different levels of participation. Ph.D. dissertation, Faculty of Educational Sciences, University of Oslo, Norway.
- Andersen, R., and A. Mørch. 2009. “Mutual Development: A Case Study in Customer-Initiated Software Product Development.” In *IS-EUD 2009. LNCS, vol. 5435*, edited by V. Pipek, M. B. Rosson, B. de Ruyter, and V. Wulf, 31–49. Heidelberg: Springer.
- Andersen, R., A. I. Mørch, and K. T. Litherland. 2021. “Learning Domain Knowledge Using Block-Based Programming: Design-Based Collaborative Learning.” In *End-User Development. IS-EUD 2021. Lecture Notes in Computer Science, vol 12724*, edited by D. Fogli, D. Tetteroo, B. R. Barricelli, S. Borsci, P. Markopoulos, and G. A. Papadopoulos, 119–135. Cham: Springer.
- Batalas, N., I. Lykourantzou, V. J. Khan, and P. Markopoulos. 2021. “Reconsidering End-User Development Definitions.” In *End-User Development. IS-EUD 2021. Lecture Notes in Computer Science, vol 12724*, edited by D. Fogli, D. Tetteroo, B. R. Barricelli, S. Borsci, P. Markopoulos, and G. A. Papadopoulos, 19–35. Cham: Springer.
- Blikstein, P. 2013. “Gears of our Childhood: Constructionist Toolkits, Robotics, and Physical Computing, Past and Future.” *Proceedings of the 12th International Conference on Interaction Design and Children (IDC’13). Association for Computing machinery, New York, NY, USA*, 173–182.
- Bocconi, S., A. Chiocciariello, and J. Earp. 2018. The Nordic Approach to Introducing Computational Thinking and Programming in Compulsory Education. Report prepared for the Nordic@ BETT2018 Steering Group, pp. 397–400.
- Braun, V., and V. Clarke. 2012. “Thematic Analysis.” In *APA Handbook of Research Methods in Psychology: Vol. 2 Research Designs: Quantitative, qualitative, neuropsychological, and biological*, edited by H. Cooper, P.M. Camic, D.L. Long, A.T. Panter, D. Rindskopf, and K.J. Sher, 57–71. Washington, DC: American Psychological Association.
- Brennan, K., and M. Resnick. 2012. “New Frameworks for Studying and Assessing the Development of Computational Thinking.” Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada Vol. 1, 25.
- Bruner, J. 1996. *The Culture of Education*. Cambridge, MA: Harvard University Press.
- Corral, L., I. Fronza, and C. Pahl. 2021. “Block-based Programming Enabling Students to Gain and Transfer Knowledge with a No-Code Approach.” In *Proceedings of the 22st annual Conference on information technology education (SIGITE 21). Association for Computing machinery, New York, NY, USA*, 55–56.
- Devine, J., J. Finney, P. de Halleux, M. Moskal, T. Ball, and S. Hodges. 2019. “MakeCode and CODAL: Intuitive and Efficient Embedded Systems Programming for Education.” *Journal of Systems Architecture* 98: 468–483.
- diSessa, A. A., and H. Abelson. 1986. “Boxer: A Reconstructible Computational Medium.” *Communications of the ACM* 29 (9): 859–868.
- Echeverría, L., R. Cobos, and M. Morales. 2019. “Improving the Students Computational Thinking Skills with Collaborative Learning Techniques.” *IEEE Revista Iberoamericana de Tecnologías del Aprendizaje* 14 (4): 196–206.
- Eisenberg, M. 2005. “The Material Side of Educational Technology.” *Communications of the ACM* 48 (1): 51–54.

- Fischer, G. 1994. "Domain-oriented Design Environments." *Automated Software Engineering* 1: 177–203.
- Fischer, G. 2021. "End-User Development: Empowering Stakeholders with Artificial Intelligence, Meta-Design, and Cultures of Participation." In *End-User Development: Is-EUD 2021. LNCS, vol. 12724*, edited by D. Fogli, D. Tetteroo, B. R. Barricelli, S. Borsci, P. Markopoulos, and G. A. Papadopoulos, 3–16. Cham: Springer.
- Fischer, G., A. C. Lemke, R. McCall, and A. I. Mørch. 1996. *Making Argumentation Serve Design. In Design Rationale: Concepts, Techniques, and Use*, edited by T. P. Moran, and J. M. Carroll, 267–293. Mahwah, NJ: Lawrence Erlbaum.
- Fogli, D., P. Peroni, and C. Stefani. 2017. "ImAtHome: Making Trigger-Action Programming Easy and fun." *Journal of Visual Languages & Computing* 42: 60–75.
- Freeman, J. 1999. "Teaching Gifted Pupils." *Journal of Biological Education* 33 (4): 185–190.
- Gennari, R., M. Matera, A. Melonio, M. Rizvi, and E. Roumelioti. 2022. "The Evolution of a Toolkit for Smart-Thing Design with Children Through Action Research." *International Journal of Child-Computer Interaction* 31, doi:10.1016/j.ijcci.2021.100359.
- Hammond, J., and P. Gibbons. 2005. "Putting Scaffolding to Work: The Contribution of Scaffolding in Articulating ESL Education." *Prospect* 20 (1): 6–30.
- Hoadley, C. P. 2002. "Creating Context: Design-Based Research in Creating and Understanding." In *Proceedings of computer support for collaborative learning (CSCL)*, Boulder, USA.
- Ingalls, D., S. Wallace, Y. Y. Chow, F. Ludolph, and K. Doyle. 1988. "Fabrik: A Visual Programming Environment." *ACM SIGPLAN Notices* 23 (11): 176–190.
- Jiang, B., W. Zhao, N. Zhang, and F. Qiu. 2022. "Programming Trajectories Analytics in Block-Based Programming Language Learning." *Interactive Learning Environments* 30 (1): 113–126.
- Jordan, B., and A. Henderson. 1995. "Interaction Analysis: Foundations and Practice." *Journal of the Learning Sciences* 4 (1): 39–103.
- Kopache, M. E., and E. P. Glinert. 1988. "C2: A Mixed Textual/Graphical Environment for C." In *Proceedings of the Workshop on Visual Languages*, 231–238. Piscataway, NJ: IEEE Press.
- Kortuem, G., F. Kawsar, V. Sundramoorthy, and D. Fitton. 2010. "Smart Objects as Building Blocks for the Internet of Things." *IEEE Internet Computing* 14 (1): 44–51.
- Li, Y., A. H. Schoenfeld, A. A. diSessa, A. C. Graesser, L. C. Benson, L. D. English, and R. A. Duschl. 2019. "Design and Design Thinking in STEM Education." *Journal for STEM Education Research* 2: 93–104.
- Lieberman, H., F. Paternò, M. Klann, and V. Wulf. 2006. "End-user Development: An Emerging Paradigm." In *End-User Development*, edited by H. Lieberman, F. Paternò, and V. Wulf, 1–7. Heidelberg: Springer.
- Lipponen, L., K. Hakkarainen, and S. Paavola. 2004. "Practices and Orientations of CSCL." In *What we Know About CSCL. Computer-Supported Collaborative Learning Series, vol 3*, edited by J. W. Strijbos, P. A. Kirschner, and R. L. Martens, 31–50. Dordrecht, NL: Springer.
- MakeCode. 2021. Introducing Microsoft MakeCode. Retrieved from <https://makecode.com/blog/makecode-overview>.
- Markoff, J. 2016. *Machines of Loving Grace: The Quest for Common Ground Between Humans and Robots*. New York: HarperCollins Publishers.
- McKenney, S., and T. C. Reeves. 2018. *Conducting Educational Design Research*. London: Routledge.
- Mönks, F. J. 1992. "Development of Gifted Children: The Issue of Identification and Programming." In *Talent for the Future. Proceedings of the Ninth World Conference on Gifted and Talented Children*, edited by F. J. Mönks, and W. A. M. Peters, 191–202. Assen, The Netherlands: Van Gorcum.
- Mönks, F. J., and M. W. Katzko. 2005. "Conceptions of Giftedness." *Conceptions of Giftedness* 2: 187–200.
- Mørch, A. I., I. Engeness, V. C. Cheng, W. K. Cheung, and K. C. Wong. 2017. "EssayCritic: Writing to Learn with a Knowledge-Based Design Critiquing System." *Journal of Educational Technology & Society* 20: 213–223.
- Mørch, A. I., K. T. Litherland, and R. Andersen. 2019. "End-User Development Goes to School: Collaborative Learning with Makerspaces in Subject Areas." In *End-User Development. IS-EUD 2019. Lecture Notes in Computer Science, vol 11553*, edited by A. Malizia, S. Valtolina, A. Mørch, A. Serrano, and A. Stratton, 200–208. Cham: Springer.
- Mørch, A., and L. Zhu. 2013. "Component-based Design and Software Readymades." In *IS-EUD 2013. LNCS, vol. 7897*, edited by Y. Dittrich, M. Burnett, A. Mørch, and D. Redmiles, 278–283. Heidelberg: Springer.
- Namli, N. A., and B. Aybek. 2022. "An Investigation of The Effect of Block-Based Programming and Unplugged Coding Activities on Fifth Graders' Computational Thinking Skills." *Self-Efficacy and Academic Performance. Contemporary Educational Technology* 14 (1): 1–16. ep341.
- Pelánek, R., and T. Effenberger. 2022. "Design and Analysis of Microworlds and Puzzles for Block-Based Programming." *Computer Science Education* 32 (1): 66–104.
- Przybylla, M., and R. Romeike. 2014. "Physical Computing and its Scope - Towards a Constructionist Computer Science Curriculum with Physical Computing." *Informatics in Education* 13 (2): 225–240.
- Reichertz, J. 2014. "Induction, Deduction, Abduction." In *The SAGE Handbook of Qualitative Data Analysis*, edited by U. Flick, 123–135. London: SAGE Publications.
- Reis, S. M., and J. S. Renzulli. 2004. "Current Research on the Social and Emotional Development of Gifted and Talented Students: Good News and Future Possibilities." *Psychology in the Schools* 41 (1): 119–130.
- Renzulli, J. S. 1978. "What Makes Giftedness? Reexamining a Definition." *Phi Delta Kappan* 60 (3): 180.
- Repenning, A., A. Ioannidou, and J. Zola. 2000. "AgentSheets: End-User Programmable Simulations." *Journal of Artificial Societies and Social Simulation* 3 (3): 351–358.
- Resnick, M., J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, et al. 2009. "Scratch." *Communications of the ACM* 52 (11): 60–67.
- Resnick, M., and B. Silverman. 2005. "Some Reflections on Designing Construction Kits for Kids." *Proceedings of the 2005 conference on interaction design and children (IDC'05)*. Association for Computing Machinery, New York, NY, USA, 117–122.
- Rienties, B., B. Giesbers, D. Tempelaar, S. Lygo-Baker, M. Segers, and W. Gijsselaers. 2012. "The Role of Scaffolding

- and Motivation in CSCL.” *Computers & Education* 59 (3): 893–906.
- Root, E., W. Heuten, and S. Boll. 2019. “Maker Cards: Evaluating Design Cards for Teaching Physical Computing to Middle-School Girls.” Proceedings of *mensch und computer 2019 (MuC’19)*. Association for Computing Machinery, New York, NY, USA.
- Sari, U., H. M. Pektaş, ÖF Şen, and H. Çelik. 2022. “Algorithmic Thinking Development Through Physical Computing Activities with Arduino in STEM Education.” *Education and Information Technologies* 1: 1–21.
- Sengupta, P., J. S. Kinnebrew, S. Basu, G. Biswas, and D. Clark. 2013. “Integrating Computational Thinking with K-12 Science Education Using Agent-Based Computation: A Theoretical Framework.” *Education and Information Technologies* 18 (2): 351–380.
- Serrano-Cámara, L. M., M. Paredes-Velasco, C. M. Alcover, and JÁ Velazquez-Iturbide. 2014. “An Evaluation of Students’ Motivation in Computer-Supported Collaborative Learning of Programming Concepts.” *Computers in Human Behavior* 31: 499–508.
- Sheriff, A., R. Sadan, Y. Keats, and O. Zuckerman. 2017. “From Smart Homes to Smart Kids: Design Research for CataKit.” In Proceedings of the 2017 Conference on Interaction Design and Children (IDC’17). Association for Computing machinery, New York, NY, USA, 159–169.
- Shneiderman, B. 2020. “Human-centered Artificial Intelligence: Three Fresh Ideas.” *AIS Transactions on Human-Computer Interaction* 12 (3): 109–124.
- Silverman, D. 2005. *Doing Qualitative Research: A Practical Handbook*. London: Sage.
- Stahl, G. 2006. *Group Cognition: Computer Support for Building Collaborative Knowledge*. London: MIT Press.
- Stahl, G. 2016. “From Intersubjectivity to Group Cognition.” *Computer Supported Cooperative Work (CSCW)* 25: 355–384.
- Stahl, G., T. Koschmann, and D. Suthers. 2006. “Computer-supported Collaborative Learning: An Historical Perspective.” In *Cambridge Handbook of the Learning Sciences*, edited by R. K. Sawyer, 409–426. London: Cambridge University Press.
- Stahl, G., S. Ludvigsen, N. Law, and U. Cress. 2014. “CSCL Artifacts.” *International Journal of Computer-Supported Collaborative Learning* 9 (3): 237–245.
- United Nations Educational, Scientific and Cultural Organization (UNESCO). 2022. Artificial intelligence in education. <https://en.unesco.org/artificial-intelligence/education>.
- Utting, I., S. Cooper, M. Kölling, J. Maloney, and M. Resnick. 2010. “Alice, Greenfoot, and Scratch -- A Discussion.” *ACM Transactions on Computing Education* 10 (4): 1–11.
- Vogel, F., I. Kollar, F. Fischer, K. Reiss, and S. Ufer. 2022. “Adaptable Scaffolding of Mathematical Argumentation Skills: The Role of Self-Regulation When Scaffolding with CSCL Scripts and Heuristic Worked Examples.” *International Journal of Computer-Supported Collaborative Learning* 17: 39–64.
- Weintrop, D., D. C. Shepherd, P. Francis, and D. Franklin. 2017. “Blockly Goes to Work: Block-Based Programming for Industrial Robots.” *Proceedings 2017 IEEE blocks and Beyond Workshop (B&B)*, pp. 29–36.
- Weintrop, D., and U. Wilensky. 2017. “Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms.” *ACM Transactions on Computing Education* 18: 1. Article 3, 25.
- Wulf, V., and B. Golombek. 2001. “Direct Activation: A Concept to Encourage Tailoring Activities.” *Behaviour & Information Technology* 20 (4): 249–263.
- Yang, S. J., H. Ogata, T. Matsui, and N. S. Chen. 2021. “Human-centered Artificial Intelligence in Education: Seeing the Invisible Through the Visible.” *Computers and Education: Artificial Intelligence* 2, doi:10.1016/j.caeai.2021.100008.
- Zhang, L., and J. Nouri. 2019. “A Systematic Review of Learning Computational Thinking Through Scratch in K-9.” *Computers & Education* 141: 103607.