Assessment strategies for cross-curricular programming in secondary education

Ana Fuentes-Martínez[1][0000-0002-1748-8837], Justyna Szynkiewicz[2][0000-0001-9118-0059], Majid Rouhani[2][0000-0002-9083-4409], and Kateryna Osadcha[2][0000-0003-0653-6423]

[1] University West, Trollhättan, Sweden `ana.fuentes-martinez@hv.se`
[2] Norwegian University of Science and Technology, Trondheim, Norway
{`justyna.szynkiewicz, majid.rouhani, katheryna.osadcha`}`@ntnu.no`

Abstract. The increasing integration of computer science and programming into formal school education is a commendable endeavor that has seen different implementation solutions. Sweden and Norway have opted for a cross-curricular model, incorporating the task of teaching and learning computing into already existing subjects, mainly within STEM modules. In-service teachers often struggle with teaching programming effectively and integrating acquired programming knowledge into their educational settings. Additionally, instructors need to understand and evaluate programming learning outcomes, taking into account the new curriculum requirements. There is a lack of clear guidance regarding how teachers could assess students' knowledge and skills when programming becomes a part of their subject. This study investigates the assessment approaches of in-service teachers who have undergone a university-level professional development program.

The qualitative analysis of the teachers' assessment plans reveals that traditional assessment strategies are adjusted for the sake of programming, leaning towards formative initiatives featuring discussions, presentations, and student projects, and to a lesser extent, tests and exams. With respect to programming, teachers' assessment initiatives cover a broad spectrum of knowledge with different degrees of abstraction and granularity, from the particularities of coding and debugging to more abstract issues of algorithmic thinking and even program quality such as robustness or reliability. Higher Education courses addressing teacher professional development in programming might therefore integrate these strategies to support teachers' assessment in programming.

Keywords: Programming, Computational Thinking, · Cross-curricular Assessment · Subject integration · K-12 · ISCED2 · ISCED3.

1 Introduction

School policies in many countries have been updated to include digital proficiency, computational thinking (CT), and programming in recognition of the significance of digitalization in modern society. Existing literature shows that teaching and learning programming can be arduous and time-consuming (e.g. [11,1]). In-service teachers, particularly those with limited time and prior programming knowledge, often struggle to teach programming effectively and integrate their newly acquired programming knowledge into their educational settings. This means not only transferring programming skills and knowledge into their actual teaching practice but doing so within the curricular constraints of their subject. Along these conditions, educators and researchers need to establish reliable measures to evaluate students' learning outcomes including fundamental programming concepts and skills.

The 2020 revision of the Norwegian curricula LK20[1] for ISCED0-3[18,10] focuses on digital skills alongside traditional basic skills like reading and writing to be emphasized in all subjects. Programming is included in the curriculum for Mathematics, Science, Music, and Arts and Crafts, with varying levels of detail given in each subject's competence goals. The curriculum for mathematics in particular, states that students shall be able to use programming to explore and solve mathematical problems whereas in science, students are instructed to "explore, construct, and program technological systems with interdependent parts" (ages 10-12) and to "utilize programming to investigate natural phenomena" (ages 13–16). This perspective of programming as a tool is also found in the curriculum for Music, Arts and Crafts, where students are encouraged to incorporate programming into their creative processes, such as using it to create interactive and visual expressions.

Modern learning theory makes clear that expertise is developed within specific domains and learning is situated within specific contexts where it needs to be developed and from which it must be helped to transfer [17]. This cross-curricular approach brings attention to the issue of integrated assessment, which is the focus of this study. As an essential part of teaching and learning with programming, assessment practices need to be acknowledged and understood in the development of teacher professional programs. Therefore, in this paper, we investigate how teachers envision assessment in their plans to implement programming activities in their respective subjects.

The following research questions (RQ) were asked in the study:

1. *Which strategies do teachers envision in their assessment of students' programming activities in Norwegian schools?*
2. *What kind of programming concepts and skills do the teachers plan to assess in this context?*

The questions address the lack of research regarding cross-curricular assessment with programming in secondary education. In order to provide teachers with the necessary guidance towards the implementation of an integrated curriculum, the purpose of this paper is to surface and critically discuss the problems and opportunities that arise when subject assessment incorporates programming.

[1] Kunnskapsløftet 2020 https://www.udir.no/laring-og-trivsel/lareplanverket

The paper is organized into seven sections. The next section situates the investigation within the ongoing research on assessment, particularly with respect to programming in secondary education. Section 3 describes the process of gathering and analyzing data for this study. The results are presented and examined in sections 4 and 5 and their implications are further discussed in section 6. Section 7 finalizes the argument with overarching conclusions.

2 Background

Assessment is a crucial teaching activity because it provides insights into students' understanding, allowing for feedback on their progress and identifying areas of improvement. Additionally, assessment can serve as a means of recognition and comparison, ensuring that learning goals are being met and preparing for future academic or career endeavors [19].

Effective teaching and learning require a close alignment between the intended learning outcomes, the learning activities and assessment methods, and the content of a course (e.g. [2,5]). Thinking in terms of alignment leads to a more focused and effective learning experience, as students are able to see the connections between what they are learning, how they are learning it, and how they are being evaluated. When teaching includes elements of programming and automation, these too shall be included in the corresponding assessment to convey coherence within the subject.

It is nevertheless difficult to isolate assessment from the rest of the activities that constitute teaching and learning. For the sake of this investigation, assessment refers to those activities initiated by the teacher with the purpose of providing information on students' achievements with respect to the teaching goals. It leaves out other perspectives on assessment, such as those aiming at curriculum development, school accountability, or teacher performance. Rather, it centers on the micro context of student work that leads the teacher to evaluation, judgment, or decision on student progress and ability [19].

Particularly entangled with assessment seems to be the concept of feedback, where the information retrieved from assessing students' achievements is returned to them in order to give the student insights into their strengths and areas for improvement [23]. Winstone and Boud mean that assessment is the process of evaluating a student's performance, while feedback is the information given to the student to help them understand their performance and make improvements [23]. Assessment generates data that informs educators about students' progress, while feedback is a tool for improvement based on that assessment data, making learning visible [16].

2.1 Assessment in Secondary Education

Providing a succinct summary of assessment practices in secondary education has proven to be a laborious task. Hubwieser et al. [9] point out that the existence of many different educational systems, with different types of schools, varying even within a single country and over time not only adds to the difficulty of building on earlier work but obstructs a unified view on assessment [21].

Concerning the purpose of assessment, scholars distinguish summative and formative assessment [4]. Summative assessment is generally given at the end of a lesson, semester, or school year to "sum up" what the student knows and has learned. It is used for evaluative purposes. On the other hand, Formative assessment is used during the learning process in order to determine gaps in a student's knowledge and to adjust instruction accordingly [4, p. 36]. To this distinction, Caffrey adds Predictive assessment, designed to determine whether a student is on track for meeting achievement goals, and Diagnostic assessment, which purpose is to determine a student's academic, cognitive, or behavioral strengths and weaknesses, and to identify students as eligible for special education. For the purpose of this investigation, we consider assessment used to determine what the student has learned rather than to plan instruction, predict future achievement, or diagnose cognitive abilities.

2.2 Assessment in Programming Education

Much of the research on assessment of programming knowledge has been conducted in Higher Education with a clear focus on automatic assessment tools of evaluation and feedback (e.g. [24]). For younger students, assessment is generally less formalized, particularly when the curriculum guidelines are vague and the progression is not well defined.

Addressing programming and CT, Werner et al. [22] proposed an assessment model called Game Computational Sophistication (GCS) suitable for younger students and Moreno-León et al. constructed an analytical tool to assess progression in terms of coding and CT skills assigning complexity scores to block-programming solutions [14]. Also, Brennan and Resnick [3] proposed a model for measuring CT skills. They suggested a framework with three dimensions; Computational Concepts, Computational Practices, and Computational Perspectives. Computational Concepts include sequences, loops, parallelism, events, conditionals, operators, and data. The Computational Practices involve focusing on the thinking and learning process while Computational Perspective is about children's "understandings of themselves, their relationships with others, and the technological world around them" [3].

The Norwegian curriculum (LK20) offers some guidelines for assessment. For example, the ordinance 'Competence aims and assessment' for mathematics (courses 1T and 1P in ISCED3) indicates that

> *The teacher and the pupils shall engage in dialogue about their development in programming and strategies for solving problems. The pupils shall have the opportunity to try and fail. With the competence the pupils have demonstrated as the starting point, they shall have the opportunity to express what they believe they have achieved and reflect on their development in the subject. The teacher shall provide guidance on further learning and adapt the teaching to enable the pupils to use the guidance*

provided to develop their competence in discovering relationships between mathematics and theoretical [or practical in 1P] *applications.*

The curriculum favors a student-centered view of assessment in which both assessment strategies and feedback are mentioned, as well as self-reflection opportunities. In this respect, it is unclear what programming content is to be included in the assessment, leaving it open for teachers to interpret and implement the guidelines.

2.3 Cross-curricular assessment

Cross-curricular skills are defined as general skills that can be taught and practiced in curricula across different disciplines [13]. From the early days of school computers, programming has been seen as a cross-disciplinary method that could foster learning in other subjects [15].

The present curriculum renews this conception of programming embedded throughout the school experience. Research in teaching and learning with programming has since then produced relevant results regarding cross-disciplinary uses of programming in other subjects. Several studies that consider the transfer benefits of learning programming, i.e., analyzing how learning programming fosters learning in other subjects (e.g. [8,12]).

There is however a crucial difference between learning another subject by means of programming and demonstrating acquired knowledge in another subject using programming. Overall, assessment in an integrated curriculum is not without challenges, stemming not only from the variation within and across the subjects in which programming can be implemented but also from the learning progression that directs the different knowledge areas [6]. When examining a cross-curricular initiative in Northern Ireland, Greenwood found that not only lack of progression but also the difficulties in assessment were among teachers' major concerns with the integrated curriculum [7, p. 447]

In a cross-curricular setting, when programming is used to demonstrate knowledge on other subjects, deficiencies in programming skills might reflect on poor results in the subject being assessed, thus undermining the conditions of fair and relevant assessment [13]. These challenges and the lack of research on assessment, when programming becomes an integral part of other subjects, motivate the following investigation.

3 Method

This section presents the details on how data was gathered and analyzed in search of cues that could shed light on the issue of assessment in programming projects. We start by describing the course in which we conducted the study together with the participants who provided the necessary information and conclude with the particulars of the thematic analysis.

3.1 Course settings

During the spring term of 2023, 148 in-service teachers from secondary schools in Norway participated in a commissioned stand-alone 7.5 ECTS credit course on Applied Programming (IT6204) at the Norwegian University of Science and Technology. The course was open for teachers who had previously taken the introductory programming course or otherwise acquired equivalent knowledge in programming constructs, data structures, and methods for testing and troubleshooting simple programs.

The participants were asked to prepare a teaching plan that incorporated programming. In the teaching plan reports, the teachers described the competence goals of the subject, the learning objectives of the teaching plan, the assessment, students' prior knowledge, approaches to learning, practical implementation, and a time plan. For this study, teachers' deliveries were analyzed with data from the assessment rubric. This was also the topic of one of the lectures during the course. Permission to use this data was granted by the participating teachers and the researchers obtained the necessary clearance from the NSD[3] for the data management plan.

Table 1. Distribution of teachers' programming plans according to subject and education level: ISCED2 (Ungdomsskole) and ISCED3 (videregående).

Subject area	Nr of projects		
	ISCED2	ISCED3	Total
Mathematics	31	36	67
Natural Science	15	7	22
Physics	-	11	11
Elective Programming	4	-	4
Chemistry	-	3	3
Biology	-	2	2
Information Technology and Media Production	-	2	2
Norwegian Language	-	1	1
Technology and Theory of Research	-	1	1
Automation, Computer technology and Electronics	-	1	1
Introductory Python course for mathematics	-	1	1
Arts, Crafts and Design	1	-	1
Physical Education	-	1	1
Total	51	66	117

[3] Norwegian Centre for Research Data

Table 1 presents the subjects in which programming was to be implemented according to the teachers' plans. The participants were allowed to collaborate with their colleagues on the teaching plans. Projects that involved several disciplines, such as those combining Mathematics and Natural Science, are counted in each of them. This accounts for the discrepancies in tallies ($148 > 117 > 98$). The projects were evenly distributed across level of education (51:66) with STEM subjects overly prevalent in both. From the 98 reports received, as many as 67 projects focused on implementing programming in mathematics.

3.2 Data analysis

We conducted a thematic analysis of the teachers' plans focusing on how assessment was envisioned. Our approach followed a systematic framework described in [20], which involves an exploratory bottom-up approach.

First, we immersed ourselves in the 98 teacher plans to become familiar with the projects. Focusing on the section of the assignment that reported on assessment, we distinguished two initial lines of analysis. From a pedagogical point of view, we considered the assessment instruments that the teachers found relevant for their programming projects while from the perspective of programming knowledge, the focus was on which programming concepts and skills were included in the assessment.

This distinction helped us to capture meaningful segments, and as we progressed, we identified recurring patterns that led us to establish category clusters. We started by looking at a few examples together and after that decided to continue individually where one researcher focused on programming concepts and the other on the strategies of the assessment. We defined each theme within the category clusters, creating a clear framework for interpretation. Finally, we cross-validated each other's categories to ensure objectivity and reliability.

We use 'codes' (plural) to refer to the entries that we assigned to the ideas in the assessment section of the teacher's work during the process of classifying the information. This needs to be distinguished from other uses of 'code' or 'coding' that also appear in the paper and are specific to computer programming, i.e. writing code or debugging code.

The following sections summarize the findings of this study structured as envisioned assessment strategies and programming content.

4 Envisioned assessment strategies

The first question guiding the investigation asks about the assessment strategies that the teachers plan to include along with the programming activities that they design in their respective disciplines. The question emphasizes the models and methods by which the students will demonstrate their knowledge. It is therefore an inquiry on *how* cross-disciplinary assessment is envisioned when programming is part of another subject's curriculum. The codes reflecting these assessment strategies (Table 2) are organized into four categories: Teacher Response, Deliverables, Reflective Assessment, and Task Assessment.

4.1 Teacher Response

In this category we find codes traversing the dimension of formative-summative assessment. Much of the teachers' plans were devoted to explaining how assessment could help students to understand their own learning and codes relating to how students get information about their development could be found in most excerpts under the assessment rubric. The teacher's input is central in this category. Most teachers believed in guiding and providing feedback for students' programming learning, while only a few planned on giving grades or grading based on goal achievement.

The notable gap between projects prioritizing guidance and feedback versus grades and goal achievement might be linked to curriculum recommendations. The curriculum emphasizes teacher-pupil dialogues and teachers' guidance for competence development in programming. Winstone and Boud claim that, although feedback and assessment are distinct concepts, they are closely interconnected [23]. Our findings align with this observation, as teachers emphasized guidance and feedback as the foremost assessment approach, rather than other subsequent actions taken after evaluating students' work.

4.2 Deliverables

This category refers to the various modes of presenting student work, such as presentations, reports, products, and showcases. It captures the idea that students can manifest their knowledge and skills in different formats, allowing for more creativity and flexibility in assessment. This category is about the diversity of methods for evaluating student performance beyond traditional tests and exams. In their framework for assessing CT with programming, Brennan and Resnick advocate for the integration of artifact assessment because it provides tangible and context-rich examples for analysis [3]. Accordingly, teachers favor this type of deliverables as instruments of assessment.

4.3 Reflective assessment

Many of the descriptions in this category encourage students to reflect on their learning experiences, challenges, and achievements. This reflective process promotes metacognition and self-awareness, helping students identify areas for improvement and growth. They provide a more holistic view of a student's learning in which the teacher's role is less prominent, letting the students assume more responsibility for their learning by being involved in the assessment process. This resonates well with Brennan and Resnick's [3] computational perspective as one of the dimensions of assessment (see Section 2.2).

4.4 Task assessment

This last category features assessment strategies in which the task is central, either as short exercises or longer assignments. The tasks are designed to offer a comprehensive evaluation of students' skills and to assess knowledge retention. When conducted in the classroom environment, task assessment allows for real-time monitoring and assistance. Task assessments in the form of tests and exams are seen by teachers as valuable practice prior to the National Exams. Finding little guidance in the curriculum, the teachers might design assessment tasks by replicating those in the National Exams where Python programming is used.

5 Programming concepts and skills

When the teachers wrote about how they intended to assess their students, their plans pursued a wide range of tasks, concepts, and methods. Among those descriptions, we examine here the statements which were explicitly related to programming. This analysis aims to inform the second research question, i.e. *what* are the programming concepts and skills that the teachers intend to assess according to their teaching plans.

The assessment ideas that the teachers proposed concerning programming were classified into four categories, representing four dimensions of knowledge and skills: Applications, CT. Program quality and Coding. Rather than suggesting different levels of difficulty, these dimensions show possible perspectives with different degrees of abstraction and granularity (see Table 3).

5.1 Applications: Programming as a Tool

This category encompasses codes that acknowledge the power of programming to organize data and extract information. Students are asked to reflect on the convenience of using programming for different tasks and even to be able to determine whether programming can be a feasible strategy to solve a given problem. To this category belongs also the ability to select among various programming paradigms and reason about their advantages for different purposes. In the case of mathematics, the curriculum states that students shall be able to choose among different solving strategies, with or without programming, which further reinforces this view.

Several teachers emphasize the necessity of automation to handle large datasets and the prevalence of programming solutions to tackle problems in real life. The teachers provide opportunities for the students to demonstrate the results of science experiments computationally, with self-generated plots or statistic analysis.

Assessment in this category is mostly carried out in the form of products and reports whereas tests and shorter exercises can prove difficult for this purpose. Nevertheless, it is not straightforward whether statements such as "I want the students to see the usefulness of programming" are readily translated into the corresponding assessment of learning, and how, in that case, this assessment would be accomplished.

Table 2. Assessment strategies

	Codes (occurrences)	Description
Teacher Response (124)	Guidance (71)	Scaffolding students' learning process. Can be given to an individual, pair, or group of students.
	Feedback (39)	Similar to guidance but more structured. Written or oral. In relation to a specific task, exercise, homework, or product.
	Goal achievement level (8)	Summative evaluation (low, medium-, or high). Indicates learner's knowledge and skills level. Assessed through for example a combination of a report, test, and presentation.
	Grading (6)	Summative assessment of the subject or programming after a concluded learning activity
Deliverables (102)	Presentation (40)	Can take many forms: oral, audio or video recording, slides. Delivered individually or in groups.
	Report (35)	A written document describing the design of the product, process of implementation, results, discussion, and quite often including reflection notes.
	Product (22)	Can take a form of a game, a program, or a design. Often a result of collaboration.
	Showcase (5)	Students demonstrate their product in the classroom. Similar to the presentation but less comprehensive, since it focuses on a specific task or product.
Reflective assessment (85)	Learning conversations (27)	Dialogue with academic content between students and the teacher where students have the opportunity to demonstrate their knowledge.
	Logbook (17)	Track-keeping journal that summarizes what students have learned. Can include screenshots, encountered problems, solutions, and reflection notes.
	Discussion (14)	Less formal and structured than learning conversations. Can involve exclusively students.
	Peer response (13)	Providing and receiving feedback from classmates
	Self-assessment (10)	Reflecting on one's work. Can be supported by assessment criteria created by teachers or in collaboration.
	Defence (3)	Students need to defend their project's outcomes.
Task assessment (60)	Exercises and assignments (34)	Programming tasks to be completed individually or in group, usually in the classroom.
	Tests (16)	To check students' programming skills. Before for diagnosis, during the activity to track progress, and after to assess knowledge retention.
	Exams (10)	Summative formal method for evaluating students' subject knowledge, usually graded, and commonly referred to as National Exams.

5.2 Computational Thinking

The teachers aim to address several of the concepts that belong to the computational thinking realm. This means that assessment concerns students' understanding of algorithms and their capacity to approach problems systematically.

Teachers resort to high-level descriptions of algorithms or a mixture of natural language and simple programming constructs to assess their students. Testing with pseudocode involves checking whether a written description accurately represents the intended algorithm or asking the students to develop a program from a pseudo-code description.

Assessment in this category includes also more abstract ideas regarding code reusability. Students are expected to know how to utilize existing code components in new programming projects. This involves understanding how to incorporate functions, classes, or libraries to avoid duplicating effort and enhance modularity.

5.3 Program Quality

At the program level, the teachers anticipate assessing the quality of students' deliveries in terms of overall performance and value. This involves evaluating factors like correctness, efficiency, maintainability, and adherence to coding standards.

Students' programming projects are assessed with regard to how they handle unexpected inputs, errors, and edge cases (Robustness) testing the program with various scenarios to ensure it behaves predictably and reliably. Assessment focuses also on the clarity and organization of a program's code such as its structure, comments, variable naming, and indentation (Readability) making it easier for other programmers to understand and maintain.

5.4 Coding skills

This category encompasses those elements of assessment that are most closely linked to the code itself. Teachers aim to estimate to which extent students are able to comprehend, manipulate, and enhance existing code. Students' abilities to work with code are assessed with respect to syntax and logic as well as in their ability to identify and rectify different types of errors, create solutions from scratch, modify existing code, and predict outcomes.

6 Further reflections

The findings above offer insights into the multifaceted nature of programming assessment, whether it is approached from the modes of assessment or the programming content.

Table 3. Teachers' assessment plans addressing programming

Categories	Related codes
Applications	usefulness, programming in mathematics, programming as a tool, modeling, simulations, problem-solving when programming is one of the possible strategies, real data, large amounts of data, plots
Computational Thinking	algorithms, algorithmic thinking, pseudo-code, blocks, functions, reuse code
Quality	robustness, readability, effectiveness
Coding	debug, syntax errors, logic errors, explain, programs, understand programs, modify programs, tinkering, extensions, copy code, predict behavior of program, PRIMM, create code from scratch, write code, functionality, suggest simplifications.

6.1 Assessment and Feedback

Assessment and feedback are frequently intertwined concepts within educational contexts, leading to a complex relationship that impacts both theoretical understanding and practical implementation. As a result, they have become entangled in both policy and practice, resulting in a conceptual and practical blurring of their unique purposes. At the onset, we intended to examine assessment by itself, but the teachers' plans with respect to this rubric revealed that providing feedback was their principal concern. Whilst the same task can result in the occurrence of both assessment and feedback, we need to pay attention to the underlying distinction between the two processes. Clear communication is required to differentiate when the focus is on evaluating performance (assessment) and when it is on guiding improvements (feedback). Assessment serves as a tool for gauging learning outcomes while feedback serves as a catalyst for growth.

6.2 Assessment of programming or assessment with programming

The teachers were asked to specify whether programming would be assessed directly or indirectly. *Direct assessment* entailed requesting students to solve a programming task while *Indirect assessment* referred to evaluating subject competence wherein programming was applied. This distinction aimed at establishing how programming could be used to assess subject content from other disciplines. However, the teachers who chose to indicate their approach understood it differently. They saw direct assessment as tasks where students were asked to write code, and indirect assessment as tasks where students were given a program and needed to explain its functionality or correct the given code. This confusion bespeaks the difficulty of even envisioning a cross-curricular assessment in which programming content is put at the service of measuring knowledge in other subjects.

7 Conclusions

The findings of this study shed light on the multifaceted nature of programming assessment in integrated curriculum settings. The analysis of the teachers' assessment plans revealed a range of strategies, including presentations, reports, exercises, and learning conversations, which highlight the importance of formative assessment and feedback in programming education. Moreover, the study identified four dimensions of programming knowledge and skills that teachers aimed to assess: programming as a tool, computational thinking, program quality, and coding skills.

Taras argues that modern theories on assessment are marked by a plethora of "unnecessary and unhelpful dichotomies [such as] formative versus summative; functions versus processes; formal versus informal assessment [. . .]" that limit the potential for understanding and development of assessment [19, p. 2]. Distinguishing between assessment strategies and programming content is not intended to add yet another division. Instead, the purpose of this analysis is to understand how teachers could integrate the strategies and the content into one single holistic assessment practice.

The results open new possibilities to include programming assessment within the course content of university-level professional development programs for teachers. By refining assessment strategies and promoting effective integration of programming into the curriculum, teacher training can help educators to prepare students for the digital age and foster their computational thinking skills.

Future studies should investigate the impact of different assessment methods on students' learning outcomes and engagement with programming. There are also implications for educational policy and practice, as highlighted by the need for clear guidance and support for in-service teachers in integrating programming into their subjects.

References

1. Abesadze, S., Nozadze, D.: Make 21st century education: The importance of teaching programming in schools. International Journal of Learning and Teaching **6**(3), 6 (2020)
2. Biggs, J.: Enhancing teaching through constructive alignment. Higher education **32**(3), 347–364 (1996)
3. Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking. In: Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada. vol. 1, p. 25 (2012)
4. Caffrey, E.D.: Assessment in elementary and secondary education: A primer. DIANE Publishing (2009)
5. Crespo, R.M., Najjar, J., Derntl, M., Leony, D., Neumann, S., Oberhuemer, P., Totschnig, M., Simon, B., Gutiérrez, I., Kloos, C.D.: Aligning assessment with learning outcomes in outcome-based education. In: IEEE EDUCON 2010 Conference. pp. 1239–1246. IEEE (2010)

6. Fuentes Martinez, A.: Teachers' tactics when programming and mathematics converge. Ph.D. thesis, University West (2021)
7. Greenwood, R.: Subject-based and cross-curricular approaches within the revised primary curriculum in northern ireland: teachers' concerns and preferred approaches. Education 3-13 **41**(4), 443–458 (2013). https://doi.org/10.1080/03004279.2013.819618
8. Guzdial, M., Dodoo, E., Naimpour, B., Nelson-Fromm, T., Padiyath, A.: Putting a teaspoon of programming into other subjects. Commun. ACM **66**(5), 24–26 (apr 2023). https://doi.org/10.1145/3587926
9. Hubwieser, P., Armoni, M., Brinda, T., Dagiene, V., Diethelm, I., Giannakos, M.N., Knobelsdorf, M., Magenheim, J., Mittermeir, R., Schubert, S.: Computer science/informatics in secondary education. In: Proceedings of the 16th annual conference reports on Innovation and technology in computer science education-working group reports. pp. 19–38 (2011)
10. Karlsen, G.E.: Kvalifikasjonsrammeverk-virkemiddel for kvalitet eller ensretting? Uniped (2010)
11. Lahtinen, E., Ala-Mutka, K., Järvinen, H.M.: A study of the difficulties of novice programmers. Acm sigcse bulletin **37**(3), 14–18 (2005)
12. Løken, T.K.: Computational thinking through geometric understanding: A case study on programming in mathematics education. In: Norsk IKT-konferanse for forskning og utdanning (2022)
13. Meijer, J., Elshout-Mohr, M., van Hout-Wolters, B.H.: An instrument for the assessment of cross-curricular skills. Educational Research and Evaluation **7**(1), 79–107 (2001)
14. Moreno-León, J., Robles, G., Román-González, M.: Towards data-driven learning paths to develop computational thinking with scratch. IEEE Transactions on Emerging Topics in Computing **8**(1), 193–205 (2017)
15. Papert, S.: Mindstorms: children, computers, and powerful ideas (1980)
16. Peterson, E.R., Irving, S.E.: Secondary school students' conceptions of assessment and feedback. Learning and Instruction **18**(3), 238–250 (2008)
17. Punie, R.V.S.K.Y.: Digcomp 2.2: The digital competence framework for citizens-with new examples of knowledge, skills and attitudes. Tech. rep., European Commission, Joint Research Centre (2022). https://doi.org//10.2760/115376
18. for Statistics, U.I.: International standard classification of education: Isced 2011. Comparative Social Research **30** (2012)
19. Taras, M.: Assessing assessment theories. Online Educational Research Journal **3**(12) (2012)
20. Terry, G., Hayfield, N., Clarke, V., Braun, V.: Thematic analysis. The SAGE handbook of qualitative research in psychology **2**, 17–37 (2017)
21. Vahrenhold, J., Cutts, Q., Falkner, K.: Schools (k–12). In: Fincher, S.A., Robins, A.V. (eds.) The Cambridge Handbook of Computing Education Research, p. 547–583. Cambridge Handbooks in Psychology, Cambridge University Press (2019). https://doi.org/10.1017/9781108654555.019
22. Werner, L., Denner, J., Campe, S.: Using computer game programming to teach computational thinking skills. Learning, education and games **1**, 37–53 (2014)
23. Winstone, N.E., Boud, D.: The need to disentangle assessment and feedback in higher education. Studies in higher education **47**(3), 656–667 (2022)
24. Zhang, L., Nouri, J., Rolandsson, L.: Progression of computational thinking skills in swedish compulsory schools with block-based programming. In: Proceedings of the Twenty-Second Australasian Computing Education Conference. pp. 66–75 (2020). https://doi.org/10.1145/3373165.3373173