Fostering Reflection in CS Teacher Education

A Video-Based Approach to Unveiling, Analyzing and Teaching Novices' Programming Processes

Johannes Fischer[1], Nora Romahn[2], and Martin Weinert[3]

[1] TU Dortmund, Dortmund, Germany johannes.fischer@cs.tu-dortmund.de
[2] TU Dortmund, Dortmund, Germany nora.romahn@tu-dortmund.de
[3] TU Dortmund, Dortmund, Germany martin.weinert@cs.tu-dortmund.de

Abstract. We focus on novice programming processes in secondary education. In Germany, prospective high-school CS teachers often face the problem that they have only very little access to actual school classrooms during their teacher preparation programmes. They only learn about teaching programming from a very abstract point of view in lectures or seminars, usually with a focus on research rather than on practice. Once they arrive in schools as graduated teachers, they are often faced with completely new situations and first have to learn how the pupils learn, program, debug, and think. To bridge this gap between university education and actual school practice, we propose a new methodology that we are currently integrating into our curriculum. Our main approach is to activate reflective processes in prospective high-school teachers by first having them watch video-material from actual programming lessons at school, and then giving them exercises that let them reflect on the material they have just seen, e.g. by asking them about possible misconceptions, or letting them speculate about the pupils' thought processes. The goal of this article is to develop a theory (supported by learning theories) on how such videos and accompanying exercises should be composed in order to activate deep reflective processes. We believe that our methodology is general enough to be applicable to a wide range of programming processes, independent of actual programming languages or paradigms, age of the pupils and students, or other external factors.

Keywords: Reflection · Video-based learning · Computing Education · Interactive learning environments · E-learning

1 Introduction

Programming is an important topic of computer science courses in school. It is not only the primary experience and fundamental activity of computer scientists, but also enables the demystification of the machine. It allows, for example, to move from being a consumer of software to become a creator thereof. Therefore it is important to provide learners with high quality programming lessons.

To ensure this quality, it is mandatory to properly educate teachers about how students learn [21,23]. As the National Research Council notes, "research on learning and transfer has uncovered important principles for structuring learning experiences that enable people to use what they have learned in new settings" [5, p.4]. An example of this is the necessity for teachers "to pay attention to students' interpretations and provide guidance when necessary." [5, p.11] Hence, teachers' knowledge about learning processes directly influences their students' learning outcomes.

An important part of this knowledge is the understanding of student behaviour, particularly what students do while coding. Its importance comes from the value that this knowledge can provide in different ways. Luxton-Reilly et al. [14] mention the prediction of students' success, identification and mitigation of students' difficulties and improvements of students' success rates through alteration of their behaviour. They also note the "significant potential to learn about student learning through analysis of coding bahaviour." [14] With these benefits in mind it is clear that teachers should be educated about what students particularly do while they code.

While the literature review of Luxton-Reilly et al. [14] focusses on studies conducted at university level, we believe that these statements hold for K–12 as well. Therefore, we propose a methodology that aims at unveiling students' programming processes and developing methods to teach these findings to prospective computing education teachers. The main questions that are guiding our research are:

RQ1 Which unexpected phenomena can be observed in programming students?
RQ2 Which theories might explain those phenomena and how are they significant for CS teachers?
RQ3 How should a university course be designed in order to foster reflection on those phenomena in prospective CS teachers?
RQ4 How can videos showing programming students be incorporated into such a course?

Because of their qualitative nature we mainly use qualitative content analysis [15] to answer the first two questions. These questions ask for the exploration of processes happening in CS classrooms. To get meaningful results, we chose to visit actual school classes, observe the pupils while programming, and subsequently describe their processes. Since *description* is based on *interpretation*, this motivates our choice of qualitative content analysis.

The last two research questions are highly different in nature and require a different approach. The main difference is that we want to develop, analyse and refine our own instructional approaches in our university courses for prospective high school teachers. To accomplish this, we use a design-based research setting [2,10,9]. This means that we first formulate the learning outcomes – as implicitly done in RQ3 – and then conduct teaching experiments aimed at achieving them. Incorporating findings from the experiments into subsequent ones and repeating this process will eventually increase comprehension of the learning processes during those courses.

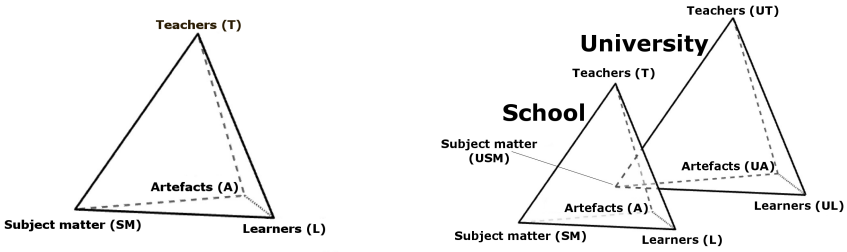
Reflection is generally agreed to be important for professional development [4] and has a long tradition of research [6,19,4,24]. There are approaches to incorporate video-based instruction into teacher training in different subjects [20,22,13,3] and even in computer science [1]. However, those approaches focus on teachers' instructional practices rather that pupils' learning processes. There are also approaches using vignettes [16] that are not video-based. To the best of our knowledge there are no previous approaches using video-vignettes of programming pupils to foster prospective teachers' reflective thoughts.

In this paper we describe our framework in detail. In Sect. 2, we define some terms that have been used liberally above. In Sect. 3, we discuss the concept of *reflection* and its significance to our project. In Sect. 4, we present the considerations serving as the basis for the video material we use. In Sect. 5, we describe our course design. Finally, in Sect. 6, we give an outline on which further steps are required to achieve the long-term goals.

2 Terminology and Structure of the Project

Because our project takes place on multiple levels of the educational system, it is reasonable to define how we refer to different actors and entities. To this end, we use the nested tetrahedron model [11]. It is a general model of academic instruction for structuring and labelling its main components.

2.1 The Didactic Tetrahedron



(a) Didactic Tetrahedron [11]

(b) Nested Didactic Tetrahedron [11]

Fig. 1: Didactic tetrahedra

The didactic tetrahedron identifies four components as important parts of learning processes in institutionalized learning settings (see also Fig. 1a). The first dimension (***L***) refers to the persons who experience the learning process. The second dimension (***T***) references the actors who facilitate the learning process. The third dimension is called subject matter (***SM***) and describes *what* the learners are learning.

The dimensions mentioned so far constitute the classic didactic triangle. What turns it into a tetrahedron is the last dimension (***A***), which considers any *thing* that serves a didactic purpose. Its importance arises from the idea that human actions manifest themselves as interactions with the world, or rather artefacts therein. (See below for concrete examples of artefacts.)

We will now model the subject of the first two research questions (RQ1 and RQ2), which refer to the school level.

Learners (L) Our main focus lies on pupils in higher secondary education, while we do not exclude other learners categorically. The important aspect with respect to our analyses is that the learners should only have little prior programming experience and work with a text-based programming language. These learners will mainly be referred to as pupils.

Teachers (T) The role of the teacher is carried out by the regular CS teacher of the class. However, since we focus heavily on *processes* happening during learning, the teacher plays a rather minor role.

Subject matter (SM) We focus on programming in general and require the learners to be active. Any task that asks for programming related activities lies in that scope, while, for example, a lecture on syntax would not.

Artefacts (A) The artefacts that the pupils will interact with depend on the concrete tasks they are working on. In general, they will always create or modify a source code, using a programming language and programming environment of some kind, using input devices such as keyboard and mouse. This means that we do not focus on *unplugged* activities, even if they involve working on program code, or on programs running on external hardware like microcontrollers or roboters.

2.2 The *Nested* Didactic Tetrahedron

With RQ3 and RQ4 we introduce the university as a second level at which we want to analyse learning processes. Since this second level might cause terminological confusion, we use an extension of the didactic tetrahedron as a conceptual framework to describe the remaining parts of our project.

Since the learning processes at university level are similar to the ones at school, the didactic tetrahedron from Sect. 2.1 would be sufficient. However, now the complete learning processes at school level will act as the subject matter at university level. This means that the tetrahedron from Fig. 1a acts as the vertex 'subject matter' of the tetrahedron at university level.

Fig. 1b illustrates the relationship of the two tetrahedra and conceptualizes the learning processes at the university level. The differentiation between the two tetrahedra is accomplished by adding a leading U to the labels. We can now describe the aspects of our project that are related to teacher training.

Learners (UL) Our focus lies on university students with the goal of becoming school teachers. These learners will be referred to as *students*.

Teachers (UT) The learning processes at university level will happen during computer science teacher training courses. Because of that the teachers (UT) are the course instructors.

Subject matter (USM) Our main goal at university level is to enable the students (UL) to reflect (see Sect. 3) on their teaching and the pupils' learning processes. To achieve that, we want them to first gain knowledge about pupils' learning processes and their associated diagnostics.

Artefacts (UA) The main artefacts that the students will use are videos showing pupils programming and exercises related to those videos. Sections 4 and 5 will explain those two artefacts in greater detail. The final goal of our project is to provide a digital learning platform that includes both the videos and the exercises.

3 Reflection

Like a computer programmer, a teacher needs to find and fix problems in his or her instructional approaches. The task of *debugging* them requires the ability of what is called *reflection*. And since assuring the efficacy of one's teaching activities is an important part of a teacher's competencies, the development of the ability to reflect on pupils' learning processes should be an integral part of teacher training curricula. As Clará [4] notes, this importance is unanimously agreed upon in the field of teacher education. However, he also notes that the degree of agreement is similar to the degree of ambiguity about what reflection actually is. The purpose of this section is not to trace the discourse on reflection, but rather find a usable definition for our project.

To find a suitable definition, we take a look at the very roots. According to Dewey [6], reflection is the

> *active, persistent, and careful consideration of any belief or supposed form of knowledge in the light of the grounds that support it, and the further conclusions to which it tends.* (p.6)

From this quote we derive the three major foci that we associate with reflection. They are (a) beliefs and (supposed) knowledge, (b) grounds, or rather reasons, and (c) derived conclusions.

Additionally, for (a) Dewey differentiates between descriptions and interpretations [4]. Here, descriptions represent the observatory part, since they are explications of (assumably) non-judgemental observations. On the contrary, interpretations are constructions of meaning. In this case, the term *constructions* means the creation of something that has not existed before, what in the case of mental constructs is denoted by *inference*.

We want the students' to reflect on all aspects of the pupils' programming processes. Beliefs and knowledge (a) will therefore be addressed by a detailed description and interpretation of the pupils' actions. A possible statement could be: *The pupil tries to declare a variable by typing the words 'Int', 'init' and 'Init', before giving up.* That statement contains an observed action (typing of words) and

its interpretations (attempt of variable declaration; resignation). Based on the given statement, a proposition for an underlying *reason* (b) might be: *The pupil seems unaware of the correct keyword for integer variables and case-sensitivity, since he inputs words similar to 'int' while varying capitalisation.* A conclusion (c) might be the proposition of an action that should be taken, like: *The pupils should be in some way instructed on the importance of correct notation of keywords.* Another possibility is a description of an alteration in the personal point of view on something: *Until I saw the pupils struggle, I didn't think that syntactical rules could be a challenge.* Finally, even general statements of (hypothetical) facts might serve as conclusions: *This example shows that the syntax of a programming language is in no way trivial to learn.*

All the given examples would be interpreted as indicators of reflective processes. To summarize what we mean by the term *reflection*, it can be described as a *reconstruction of the scientific investigation of the programming processes.* However, we do not require the students' analyses to be as rigorous as our own, but rather a small scale version thereof.

It is still open how such reflective processes might be fostered and how to support them. One answer might be suggested directly by Deweys and Schöns works: Clará [4] notes that a key aspect of reflection lies in the *clarification of incoherences* of given situations. It is therefore crucial to confront the students with examples featuring factors that lead to irritation, which finally results in the perception of incoherences.

To be able to perform such confrontations, we use video recordings of programming pupils. How these videos are composed and recorded will be described in the next section.

4 Video-Based Reflection

In the previous section we argued that we want to confront the students with their incoherent understandings in order to achieve reflection. Based on our research focus, these confrontations will be about viewing pupils' programming processes. We identify the following properties that the material (USM) should fulfil.

Authenticity: Inauthentic situations would be perceived by the students as staged. This would directly resolve any incoherences in the understanding, as the students could possibly just argue that the pupil's perceived behaviour can be attributed to the artificial situation.

Informativeness: The material needs to enable the viewers to derive reasonable and useful conclusions for themselves. Material that does not allow for statements on important topics cannot be used in education on that topic. Therefore our material in question has to provide referenceable actions of the pupils.

Anonymity: As there should be as few restrictions as possible on the usage and dissemination of the material, we want it to be practically completely anonymised so that it does not allow for the identification of the pupils.

These properties are influenced by the *Critical Incident Technique* [8,7,16]. With *informativeness* we require our videos to show *incidents*, but we do not expect them to leave the observer with as little doubt as they would need to qualify as *critical* in Flanagan's sense [8]. The properties are also influenced by the term *vignette*, as described by Jeffries and Maeder [12].

To achieve these goals, we decided to use *videos* as the primary medium. The main difference to vignettes is that our videos show real work rather than hypothetical stories. Nonetheless, we use the term *video vignettes* to refer to the combination of our videos and exercises.

Videos can meet the demands mentioned above if certain conditions are ensured. To assure authenticity we prefer recordings of pupils' regular working environments (classroom equipped with computers) over laboratory settings. We believe that this is already a quite strong criterion for authenticity. We further keep the setting for recording as little invasive as possible (see last paragraph of this section).

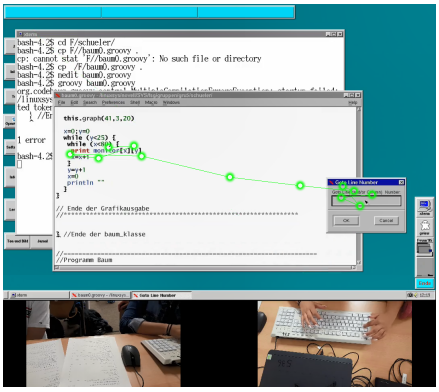
A trade-off exists between the demands on informativeness and anonymity, as anonymisation in its essence means the exclusion of certain information. An example is the removal of facial recordings, which grants non-identifiability, but at the same time prohibits observations of the pupil's emotional state. Therefore, a compromise has to be made between the demands on informativeness and anonymity. We try to accomplish this compromise by replacing the information lost through anonymisation with supplementary information. For example, the information lost by excluding facial recordings is partly reintroduced by adding video recordings of the pupils' input devices, hands and notes. These might enable the students to get a hint of emotions if they are intense enough to result in motor responses of the hands. Additionally, that view also indicates actions like pointing at the screen, which might be useful as well. Since the learners (L) often work in pairs, they are practically required to communicate verbally with each other, so we also add transcribed versions of the audio recordings.

The screen recordings can be realized in two ways: either by software running on the system that is being recorded, or by additional hardware. Since the videos will be recorded in schools and the specific features of the computer systems (hardware and software) can be quite different from school to school, we ruled out a software solution. The hardware solution does not require any software to be installed or run on the recorded systems. It consists of a video grabber that is installed between the computer and the screen and intercepts the video signal before passing it through to the screen. Altogether this provides the flexibility that is needed to record in unknown settings.

The final aspect of our videos is another supplementation used to support the interpretation of what is seen in the videos: eye-tracking data. As Przybylla [17] notes, eye-tracking data can demonstrate how someone constructs understanding of a program. While that note was made primarily on a research setting, we think that students can benefit in the same way. At the very least, eye-tracking data can be used as a rough indicator of what might be of interest for the pupils (e.g., *do they even look to the bottom right corner of the screen when a compiler error*

message appears?). Since we emphasize non-intrusiveness over precision, this led to the choice of a remote system installed in front of the monitor (instead of a head mounted one). Fig. 2a shows how a pupil's workstation looks with all our hardware installed.

(a) Workstation with recording hardware: (1) Camera recording notes, (2) Eye-tracker, (3) Framegrabber, and (4) Camera recording input devices



(b) Example of a final video.

Fig. 2: Recording hardware and video example

Fig. 2b shows the composition of the final videos. They consist of the three views discussed above: the large view at the top, showing the pupils' screen. This view also features a gazeplot that visualizes the eye-tracking data. At the bottom there are two smaller views showing the notes and input devices.

5 Course Design

We use a design-based research approach [2,10,9] to gain knowledge about the students' learning processes while simultaneously developing a course that can be used in teacher training. The design-based research process consists of cycles that repeatedly refine a design. This refinement is achieved by conducting design experiments and drawing theories from them that are used to modify the next design cycle.

First cycle

In the first design cycle we had the challenge that there were no videos available that we could use. We dealt with that issue by incorporating video production into the course. This led to a design containing six steps for the students to complete:

1. Read literature on misconceptions [18].
2. Present the literature in class.

3. Learn our specific video production processes.
4. Record videos during lessons at a school.
5. Attend an analysis session.
6. Write a term paper.

The approach has been evaluated using a written survey after course completion as well as qualitative observations during the course.

Observations First, we describe our qualitative observations during the course. Our first finding is that video production bound a lot of time in rather technical activities, like learning how to handle the recording devices or how to edit the recordings. During that time the students neither learned anything about pupils' work nor were they encouraged to perform reflective thought processes. We therefore agreed to remove video production from the course design.

During the analysis session students had to look for *interesting situations* in the videos. The term *interesting situation* had not been defined explicitly in order to find out which situations the students would perceive as interesting, but was highly influenced by their prior reading of literature on misconceptions. Apart from this, we would have liked them to identify other difficulties as well, but this did not happen. We concluded from this that students do not automatically reflect in the sense of Sect. 3 and need more fine-grained instructions.

Empirical Evaluation Next, we describe the insights we got from the survey that we conducted after the course had been completed. The survey was a voluntary online questionnaire. The students participated after their term papers were graded. Seven of the eleven course participants completed the survey. In the survey we asked them about their opinion on the relevancy of the course topics for teachers. We also asked how the course influenced those opinions and whether they learned to perform the corresponding actions (e.g. to draw conclusions based on pupils' conceptions). Additionally, we asked what activities should be removed, done less of, done more of, or be added to the course.

First, we present the reported changes in opinions during the course. Regarding the knowledge of pupils' typical conceptions and the ability to draw conclusions, about half of the participants stated that they see them more important than before the course. The other half answered that their opinion had not changed. Therefore we investigated what opinion that group had before, and found that they already strongly agree on the importance of the concepts in question.

Approximately the same distribution of opinions can be found in regards to identifying pupils' conceptions: Half of the participants see the topic more important than before and now totally agree on the importance for teachers. The other half did not change their opinion and does not totally, but mostly agree on the importance. However in this analysis we see a single student that did not change his or her opinion and mostly disagrees on the importance. Investigating that student's other answers showed that he or she seems to see no value in the programming process. We assume this because he or she agreed strongly on the

statement that videos cannot help to grade the final products in a fairer way. Additionally he or she does not agree that the work process should be observed and taken into account during grading.

This opinion raises the question if we should address the significance of the programming process explicitly during further development of our tasks. To answer that question, we investigated whether there were more students with a similar opinion. In fact we found another one that gave similar answers to those questions. In contrast to the first student, this one agrees that teachers should be able to identify pupils' conceptions by observing their work processes. This student seems not to regard the process as an integral part of programming. We will emphasize the role of the programming process more in our future tasks.

Finally we present suggestions that the participants gave for improving the course. The first comment was that the classroom visit was perceived as great, because it gave insights into future practice. It was suggested to do multiple visits and ideally do them in multiple classes. These statements support the need for a connection between the theoretical aspects of teacher training and future practice. However, classroom visits take a lot of time, which is limited during courses at the university.

Another participant reported that he or she did not feel able to draw conclusions from pupils' conceptions ((c) from Sect. 3). He or she asked to include examples of possible conclusions and explanations on how to draw them. Since this is a rather creative and especially situational process, case studies in the form of video recordings might be particularly helpful to that end. We will use frequent group discussions as the main tool to foster this process. We expect that the discussions will allow review and enrichment of suggested conclusions by other participants.

A similar point was addressed by a different participant. He or she asked to include the discussion of assistive measures and how to conduct interviews with pupils. He or she motivated that with a lack of guidance on how to deal with misconceptions. We will integrate those topics into the course. While there were presentations on how to address misconceptions, that information seems to not have been received well, as the next suggestion will show.

One participant suggested to summarize the information from the presentations of the other students. While one might argue that for a university student it should be obvious that they should take notes by themselves, we will take note of the fact that it seems not to be that obvious from the students point of view. An explanation might be that the students do not perceive the presentations of their fellow students as important with regards to their own learning goals of the course. In future iterations we should at least explicitly tell the students that we expect them to learn the topics from those presentations.

6 Conclusions and Future Work

We introduced a new methodology to foster prospective programming teachers' reflection processes. Our approach is based on video vignettes showing pupils

work on programming tasks, accompanied with appropriate exercises to encourage reflection.

Future work includes both work on school and university level. Our goal on school level is to build a larger repertoire of videos, thereby contributing to an understanding of how pupils program. The long term vision for this collection would be to identify and categorize components of programming processes that allow to induce hypotheses.

At the university level we follow our design research setting of conducting and analysing design experiments. The students' (UL) work processes and term papers will be analysed similarly to those of the pupils with qualitative content analysis [15]. The goals at this level are to learn about how students learn the specific content, how they can be supported in that and to develop concrete tasks for them.

Acknowledgements

This work is supported by the *Bundesministerium für Bildung und Forschung* under Grant No.: *16DHB2130*.

References

1. Broneak, C., Lucarelli, C., Rosato, J.: Exploring the Use of Video Reflection as a Professional Development Tool. In: Proceedings of the 2019 ACM Conference on International Computing Education Research. p. 293. ICER '19, Association for Computing Machinery (2019)
2. Brown, A.L.: Design Experiments: Theoretical and Methodological Challenges in Creating Complex Interventions in Classroom Settings. The Journal of the Learning Sciences **2**(2), 141–178 (1992)
3. Cannings, T., Talley, S.: Bridging the Gap between Theory and Practice in Preservice Education: The Use of Video Case Studies. In: Proceedings of the 3.1 and 3.3 Working Groups Conference on International Federation for Information Processing: ICT and the Teacher of the Future. CRPIT '03, vol. 23, pp. 17–20. Australian Computer Society, Inc. (2003)
4. Clarà, M.: What is Reflexion? Looking for Clarity in an Ambiguous Notion. Journal of Teacher Education **66**(3), 261–271 (2015)
5. Council, N.R.: How People Learn: Brain, Mind, Experience, and School: Expanded Edition. The National Academies Press, Washington, DC (2000). https://doi.org/10.17226/9853
6. Dewey, J.: How we think. D. C. Heath & Co., Boston (1910)
7. Finch, J.: The Vignette Technique in Survey Research. Sociology **21**, 105–114 (1987)
8. Flanagan, J.C.: The Critical Incident Technique. Psychological Bulletin **51**(4), 327–358 (1954)
9. Geldreich, K., Simon, A., Hubwieser, P.: A Design-Based Research Approach for introducing Algorithmics and Programming to Bavarian Primary Schools. Theoretical Foundation and Didactic Implementation. MedienPädagogik **33**, 53–75 (2019)

10. Gravemeijer, K., Cobb, P.: Design research from a learning design perspective. Educational design research (Jan 2006)
11. Hußmann, S., Kranefeld, U., Kuhl, J., Schlebrowski, D.: DoProfiL - Das Dortmunder Profil für inklusionsorientierte Lehrerinnen- und Lehrerbildung, chap. Das geschachtelte Tetraeder und inklusionsorientierte Designprinzipien als Modelle für Entwicklung und Forschung in einer inklusionsorientierten Lehrerinnen- und Lehrerbildung, pp. 11–25. Waxmann Verlag GmbH (2018)
12. Jeffries, C., Maeder, D.W.: Comparing Vignette Instruction and Assessment Tasks to Classroom Observations and Reflections. The Teacher Educator **46**, 161–175 (2011)
13. Kale, U., Hur, J.W., Yerasimou, T., Brush, T.: A Model for Video-Based Virtual Field Experience. In: Proceedings of the 7th International Conference on Learning Sciences. pp. 944–945. ICLS '06, International Society of the Learning Sciences (2006)
14. Luxton-Reilly, A., Simon, Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., Szabo, C.: Introductory Programming: A Systematic Literature Review. In: Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education. pp. 55–106. ITiCSE 2018 Companion, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3293881.3295779
15. Mayring, P.: Qualitative content analysis: theoretical foundation, basic procedures and software solutions. Klagenfurt (2014)
16. Pieper, U., Vahrenhold, J.: Critical Incidents in K–12 Computer Science Classrooms – Towards Vignettes for Computer Science Teacher Training. In: Proc. SIGCSE'20 (2020), to appear
17. Przybylla, M.: Programming Code Reading Skills: Stages of Development Encountered in Eye-Tracking Data. In: Busjahn, T., Schulte, C., Tamm, S., Bednarik, R. (eds.) Eye Movements in Programming Education II: Analyzing the Novice's Gaze. Proceedings of the Second International Workshop (2014)
18. Qian, Y., Lehman, J.: Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. TOCE **18**(1), 1:1–1:24 (2017). https://doi.org/10.1145/3077618
19. Schön, D.A.: Educating the Reflective Practitioner. John Wiley & Sons Inc (1987)
20. Seidel, T., Stürmer, K.: Modeling and Measuring the Structure of Professional Vision in Preservice Teachers. American Educational Research Journal **51**(4), 739–771 (2014)
21. Sentance, S., Csizmadia, A.P.: Computing in the curriculum: Challenges and strategies from a teacher's perspective. EAIT **22**(2), 469–495 (2017). https://doi.org/10.1007/s10639-016-9482-0
22. Shrader, G., Fishman, B., Barab, S., O'Neill, K., Oden, G., Suthers, D.: Video Cases for Teacher Learning: Issues of Social and Organizational Design for Use. In: Proceedings of the Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community. pp. 708–709. CSCL '02, International Society of the Learning Sciences (2002)
23. Webb, M., Davis, N., Bell, T., Katz, Y.J., Reynolds, N., Chambers, D.P., Syslo, M.M.: Computer science in K-12 school curricula of the 2lst century: Why, what and when? EAIT **22**(2), 445–468 (2017). https://doi.org/10.1007/s10639-016-9493-x
24. Zeichner, K.M., Liston, D.P.: Teaching Student Teachers to Reflect. Harvard Educational Review **57**(1), 23–48 (1987)