Applying the Mathematical Work of Teaching Framework to Develop a Computer Science Pedagogical Content Knowledge Assessment

Yvonne Kao
WestEd
400 Seaport Court, Suite 222
Redwood City, CA
ykao@wested.org

Katie D'Silva
WestEd
400 Seaport Court, Suite 222
Redwood City, CA
kdsilva@wested.org

Aleata Hubbard
WestEd
400 Seaport Court, Suite 222
Redwood City, CA
ahubbar@wested.org

Joseph Green
WestEd
400 Seaport Court, Suite 222
Redwood City, CA
jgreen@wested.org

Kimkinyona Cully
WestEd
400 Seaport Court, Suite 222
Redwood City, CA
kcully@wested.org

ABSTRACT

Pedagogical content knowledge (PCK) is specialized knowledge necessary to teach a subject. PCK integrates subject-matter content knowledge with knowledge of students and of teaching strategies so that teachers can perform the daily tasks of teaching. Studies in mathematics education have found correlations between measures of PCK and student learning. Finding robust, scalable ways for developing and measuring computer science (CS) teachers' PCK is particularly important in CS education in the United States, given the lack of formal CS teacher preparation programs and certifications. However, measuring pedagogical content knowledge is a challenge for all subject areas. It can be difficult to write assessment items that elicit the different aspects of PCK and there are often multiple appropriate pedagogical choices in any given teaching scenario. In this paper, we describe a framework and pilot data from a questionnaire intended to elicit PCK from teachers of high school introductory CS courses and we propose future directions for this work.

CCS CONCEPTS

• **Social and professional topics~Computer science education**

KEYWORDS

assessment; high school; pedagogical content knowledge; teacher preparation

ACM Reference format:

Kao, Y., D'Silva, K., Hubbard, A., Green, J., and Cully, K. 2018. Applying the Mathematical Work of Teaching Framework to Develop and Computer Science Pedagogical Content Knowledge Assessment. In *SIGCSE '18: 49th ACM Technical Symposium on Computer Science Education, Feb. 21–24, 2018, Baltimore, MD, USA.* ACM, NY, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159521

1 INTRODUCTION

Shulman introduced the idea of *pedagogical content knowledge* (PCK) as a special category of knowledge for teaching [21]. Whereas *subject-matter content knowledge* is necessary for anyone who seeks to master a subject, PCK is only necessary for those who are teaching it. Aspects of PCK include knowledge of how students think about concepts, knowledge of what makes learning concepts difficult, and knowledge of how to use different content representations to make concepts understandable for others [21].

This paper describes an effort to develop a framework for understanding computer science teaching knowledge, drawing from similar efforts in mathematics education, and to develop items that will elicit this knowledge from computer science teachers. The results reported here are a subset of the data collected for a larger project to document knowledge change in novice computer science teachers.

SIGCSE '18, February 21–24, 2018, Baltimore, MD, USA

ACM 978-1-4503-5103-4/18/02…$15.00
https://doi.org/10.1145/3159450.3159521

1.1 Pedagogical Content Knowledge in Mathematics Education

There is a large body of work in mathematics education to refine the concept of PCK and develop measures for it. This project aimed to adapt those methods into a computer science education context. The Learning Mathematics for Teaching Project [1] elaborated and refined Shulman's concept of PCK in mathematics by analyzing the work of mathematics teachers and identifying how mathematical knowledge is used. This paper focuses on four of these categories:

1. *Common content knowledge (CCK)*: the mathematical knowledge and skill that is common across disciplines that use mathematics, such as performing calculations or otherwise solving mathematics problems.
2. *Specialized content knowledge (SCK)*: the mathematical knowledge and skill unique to teaching, such as evaluating whether a student's non-standard approach to a problem is sufficiently general.
3. *Knowledge of content and students (KCS)*: knowledge that combines knowing about students and knowing about mathematics, such as knowing what students will find motivating or confusing.
4. *Knowledge of content and teaching (KCT):* knowledge that combines knowing about teaching and knowing about mathematics, such as understanding how to sequence topics or choose representations for instruction.

Hill and colleagues [8] then used this framework to develop assessments of Mathematics Knowledge for Teaching (MKT). The MKT focuses on teachers' knowledge of content (both common and specialized) and knowledge of students and content. Teachers' performance on these assessments was found to predict student learning gains in first and third grades, after controlling for students' absence rates and demographic characteristics, classroom characteristics, and teachers' professional backgrounds [9].

1.2 Pedagogical Content Knowledge in Computer Science Education

The concept of PCK has recently been applied to CS education research. For example, in a study of experienced computing teachers learning to teach a new programming paradigm, Liberman et al. [12] explored the connection between CS PCK and content knowledge. They found teachers entered a state of *regressed expertise* where they displayed elements of both novice teaching and expert teaching. Buchholz et al. [4] provided anecdotal evidence that pre-service CS teachers progress through stages of complexity in their PCK when developing teaching modules. Baxter's [2] case study of two experienced CS teachers, one formally trained and the other self-taught, demonstrated how PCK could vary across teachers with similar levels of expertise and how PCK was enacted differently in their classrooms. Lastly, the KUI research project [3] created a framework of CS PCK that focuses on both content knowledge and the processes of teaching.

CS education researchers have employed a variety of methods to study PCK. For example, several scholars used the CoRe (Content Representation) instrument to elicit educators' ideas about teaching CS [4,7,13,18]. Interviews have been used to understand instructional strategies, the impact of lessons on teacher knowledge development, and how teachers judge the difficulty of topics for their students [2,11]. Ohrndorf and Schubert [15,16] developed an assessment that asked teachers to explain possible student responses and misconceptions. Saeli et al. [19] employed a document analysis approach to analyze the PCK information contained in CS textbooks. Doukakis and colleagues [6] developed an assessment for in-service CS teachers based on the Technological Pedagogical and Content Knowledge (TPACK) framework [1] that included two items targeting PCK [14]. In another paper, we described the development of think-aloud tasks we created to drive discussion about teachers' PCK [10]. While scholars have employed a variety of methods to study and define PCK in CS, the existing body of research does not provide strong empirical evidence linking specific pedagogical practices to student learning outcomes [17]. Furthermore, there has yet to be a validated assessment of CS PCK that can be used to evaluate teachers' preparedness to teach an introductory CS course.

This lack of validated assessments of CS PCK presents a challenge for evaluating the numerous efforts around the United States to train more K-12 computer science teachers, especially given the lack of formal CS teacher preparation and certification programs [5]. How can we evaluate the PCK content of such professional learning programs? How can we tell when a teacher is equipped to teach computer science? A validated CS PCK assessment could help us address these questions and better gauge the community's efforts at increasing the CS teacher workforce.

2 DEVELOPING AN ASSESSMENT OF CSPCK: FRAMEWORK AND CONTENT

Developing measures of PCK that can be used to evaluate individual teachers is a challenging proposition—even the Mathematics Knowledge for Teaching assessments are intended only for assessing groups of teachers for research purposes [20]. In conducting this preliminary foray into creating such an assessment, we were inspired and guided by the Mathematical Work of Teaching framework [20]. Selling and colleagues [20] identified four areas of difficulty for PCK assessment developers and created a framework for navigating them. The four areas of difficulty are:

1. Developing items that measure SCK instead of exclusively measuring CCK,

[1] TPACK is another extension of Shulman's categories of teacher knowledge that incorporates teacher knowledge of technology and how to teach with technology.

2. Developing items that measure SCK without touching on pedagogical decision-making,
3. Identifying the work of teaching that draws on teachers' SCK, and
4. Understanding how SCK might be used across multiple grade bands.

Selling et al.'s Mathematical Work of Teaching (MWT) framework can be used to construct measures of teachers' subject-matter knowledge, especially specialized content knowledge. The MWT framework is organized around three mathematical objects: explanations (including justifications and reasoning), mathematical structure, and representations. Each object is aligned to a set of actions (e.g., comparing explanations, critiquing explanations, or writing explanations). Assessment developers can then write prompts for the different actions. We follow a similar structure for our CSPCK assessment framework, though we also deviate from it in important ways. In adapting the MWT framework for CS, we consulted and received feedback from two experts in computer science pedagogy before arriving at our final constructs and teaching actions.

Our framework is organized around two areas of teacher knowledge: 1) knowledge of **explanations and representations** and 2) knowledge of **suboptimal solutions, bugs, and misconceptions**. We combine explanations and representations, which are separate mathematical objects in the MWT framework, into one category because explanations often include one or more representations. The actions associated with knowledge of explanations and representations include developing, delivering, and critiquing explanations for a given CS concept. Items aligned to this first area were expected to elicit more responses related to CCK and SCK, but responses could also touch on KCS or KCT. The second area in our framework, knowledge of suboptimal solutions, bugs, and misconceptions, is related to the mathematical structure area in the MWT framework, in that assessment items may ask teachers to analyze the structure of student work. However, we chose to broaden the construct to increase the likelihood of capturing elements of KCS and KCT—the MWT framework is largely focused on eliciting SCK. The actions associated with this second area include predicting student errors and evaluating student work. Items aligned to this second area were expected to elicit more responses related to KCS, but would likely also invoke CCK, SCK, and possibly KCT.

2.1 Item Development

We developed a 9-item preliminary assessment divided into three sections, each covering a different content area that is fundamental to introductory computer science courses: algorithms, variables and assignment, and control structures. Each question was presented as a brief teaching scenario that required the respondent to engage in a real-world teaching task. All questions were open-ended. An experienced high school computer science teacher reviewed the draft questions for interpretability. The first question in each section asked respondents to identify what was most important for students to understand about the topic during an introductory lesson. The inclusion of this question was inspired by the CoRe instrument [13]. Then each section contained two additional questions: one which targeted knowledge of explanations and representations and one which targeted suboptimal solutions, bugs, and misconceptions. Table 1 describes the prompts.

Table 1: Alignment of Questions to Areas of Teacher Knowledge

#	Target	Description
Algorithms		
2	Suboptimal solutions, bugs, misconceptions	Given a student-written algorithm for making a sandwich, describe how to highlight important ideas about algorithms while following the directions to make a real sandwich.
3	Explanations / representations	Discuss the difficulties or limitations of the sandwich activity for introducing algorithms to introductory CS students.
Variables and Assignment		
2	Explanations / representations	Decide if "A *variable* in programming is a name that refers to information the program can use later" is accurate. If not, explain what is misleading and how to improve the definition.
3	Suboptimal solutions, bugs, misconceptions	Answer a homework question about variables and assignment, predict students' incorrect responses on the question, and explain why students make those errors (see Appendix).
Control Structures		
2	Suboptimal solutions, bugs, misconceptions	Diagnose a student's level of understanding of control structures based on a code segment she wrote for an assignment. The code segment contains many repeated lines of code and no loops.
3	Explanations / representations	Describe feedback that would improve the student's understanding of control structures.

3 RESEARCH METHOD

3.1 Participants

The results reported are from 21 high school computer science teachers. Nine were teaching a pre-AP introductory computer science course using a block-based programming language; 12 were teaching Advanced Placement Computer Science A in Java. Four teachers had degrees in computer science. All were participating in an in-service professional learning program in which industry professionals co-teach with high school teachers to launch computer science programs at their schools. At baseline, participants averaged 1.14 years with the program, 1.5 years teaching CS, and 10.3 years teaching overall.

3.2 Procedure

3.2.1 Data collection. Participants completed the online CS PCK assessment at the beginning and end of the school year as part of a larger survey. Only teachers who completed both pre-test and the post-test are included in this analysis. When questions included code segments, the pre-AP course teachers were shown Snap! blocks while the AP CS A teachers were shown Java code. Participants were instructed to take no more than 30 minutes to complete all nine questions.

3.2.2 Coding. Participants' responses to the questions were coded to determine if the items were eliciting the types of teacher knowledge expected. The codes were based on Ball's domains of mathematical knowledge [1]. A single response could receive multiple codes. Specialized and common content knowledge were condensed into one category, content knowledge (CK), due to difficulty distinguishing between specialized and common content knowledge in responses. Responses that gave vague references to content were not coded as CK (i.e., "It is important for students to learn what is an algorithm"). The KCT and KCS codes were applied using Ball's descriptions (i.e., "Variable is simply a holding place. I refer to it as a bucket and use visuals to demonstrate" was coded KCT; "Students don't have much trouble generating if-then statements, but they don't often think of ways to use other varieties of conditional statements, loops, and nested loops" was coded KCS). KCT and KCS codes were not applied to more general knowledge of pedagogy or student learning unless the response explained how the concept applied to computer science (i.e., "Do a lot of examples. Do a few problems together"). The third question about variables was not coded, as all answers responsive to the prompt would have been coded KCS.

Raters first discussed the codes and coded a sample of 30 pre-test responses for the variables questions to calibrate[2]. If there was inter-rater disagreement, researchers discussed and came to consensus on the appropriate code or codes to assign. Then two raters coded the full data set, re-coding the initial 30 responses used for calibration. Approximately 50 responses were coded by both raters to check for inter-rater reliability. The average Cohen's kappa for the algorithms section was 0.58. The average Cohen's kappa for the variables and control structures sections was 0.76 and 0.78, respectively. When there was inter-rater disagreement, the lead rater made a final determination on the appropriate code to assign.

4 RESULTS

4.1 Types of Teacher Knowledge Elicited

An analysis of the codes assigned to teacher's responses revealed that the prompts did elicit different types of teacher knowledge.

4.1.1 What's important to learn in this lesson? The first question in section, which asked teachers to describe what was most important about the topic for an introductory lesson, largely elicited CK across all three topic areas—nearly all 21 teachers in the sample provided a response that was coded as CK. There was little change from pre-test to post-test (see Fig. 1).

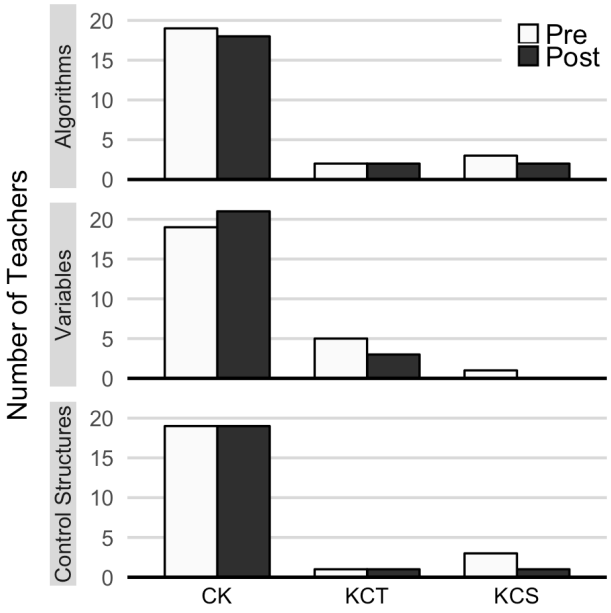


Figure 1: Teachers mainly demonstrated CK when responding to questions about what was important for students to learn about a topic in an introductory lesson.

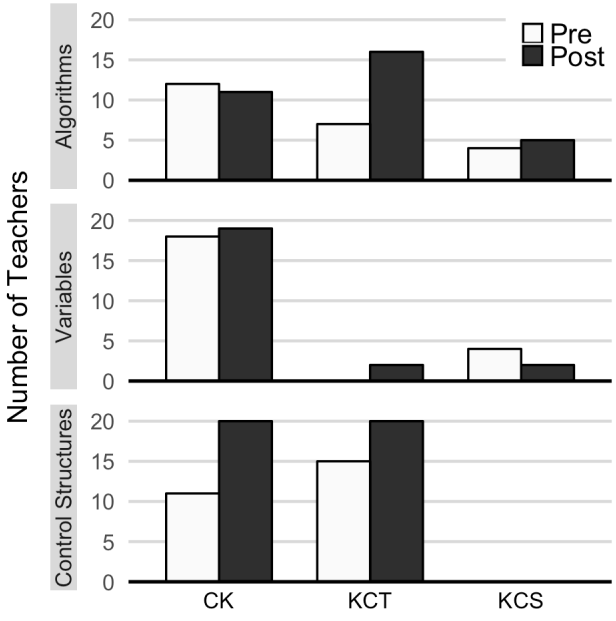


Figure 2: Teachers demonstrated some growth from pre-test to post-test on knowledge of explanations and representations.

[2] The data reported in this paper represent a subset of all data collected for the project. Rater calibration was conducted on the larger data set.

4.1.2 Teacher knowledge of explanations and representations. There is a different pattern of responses for each question in this area, probably because the three questions used qualitatively different teaching scenarios. The algorithms and control structures questions elicited CK and KCT; the algorithms question also elicited a small amount of KCS. There was a noticeable change from pre-test to post-test on the algorithms and control structures questions. The response pattern to the variables question is like that seen in the "what's important?" questions—nearly all teachers in the sample provided a CK response, with little change from pre-test to post-test (Fig. 2).

4.1.3 Teacher knowledge of suboptimal solutions, bugs, and misconceptions. The two questions that were aligned to this area asked teachers to react to student work. The algorithms question elicited mainly CK and KCT while the control structures question elicited mainly CK and KCS (Fig. 3).

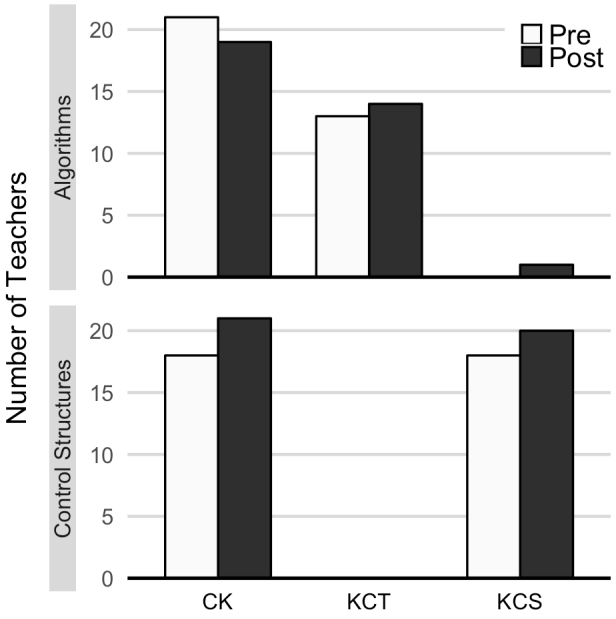


Figure 3: Suboptimal solutions, bugs, and misconceptions: the algorithms question elicited mainly CK and KCT while the control structures question elicited CK and KCS.

4.2 Teachers' Predictions of Student Errors

Table 2: Most Common Errors Predicted by Teachers

	# Teachers Predicting Error	
Error	Pre-test	Post-test
15	12	12
5	11	11
10	4	5
22	4	1

The third question in the variables section asked teachers to answer a homework question and predict student errors on the question (see Appendix). No teachers in this analysis answered the question incorrectly on pre-test or post-test. Table 2 shows the four errors most commonly predicted by teachers—the distribution is largely the same from pre-test to post-test.

Three of these errors are reflected in Sorva's "misconception catalogue" [22]; Sorva does not indicate which might be more common. Students who do not understand the sequentiality of statements may say the output is 15 because they use y = 10 when evaluating x + y, or they may say the output is 5 because they ignored the other lines of code. Or a student with the parallelism misconception may think the output is 22 because they think the line x = x + y is always active.

Many teachers predicted 10 as a student error, citing students' "carelessness" or "confusing x and y" rather than a content-oriented misconception.

5 DISCUSSION

5.1 Limitations and Lessons Learned

5.1.1 Face validity of prompts. The algorithms questions were problematic, as demonstrated by the low inter-rater reliability. The scenario presented in this section elicited many responses that were more related to classroom management than CS (e.g., a limitation of the activity is that it makes a mess, or students may have food allergies). The control structures questions showed better inter-rater reliability, but did not necessarily elicit rich responses. Many teachers diagnosed that the student did not understand looping well, but then gave simple directives to change the code (e.g., "I would advise Alice to use a loop to minimize her coding") instead of conceptually-oriented feedback.

5.1.2 Generalizability. This study cannot yet inform our understanding of how PCK might differ between expert and novice computer science teachers, partly because the prompts need refinement and partly because the participants consisted almost entirely of novice computer science teachers in a single PD program. We also did not code for correctness of the content knowledge demonstrated. However, we still found evidence of differing levels of teacher knowledge. For example, although the following responses to the third algorithms question were both coded as KCT, the second response, which was provided by a technology professional in the larger study, demonstrates deeper understanding of the limitations of the exercise.

1. I think it provides a basic understanding of the specificity necessary for computers.
2. The limitation of this exercise is that it's limited to making a physical product and the set of instructions you can specify is infinite (as opposed to the finite set of instructions a computer can execute). It's difficult to guide students to give instructions at the right level of abstraction since it's hard to describe the interface of human capabilities.

5.2 Conclusions and Future Directions

The development and testing of this short preliminary assessment provided useful insights. Participating teachers did

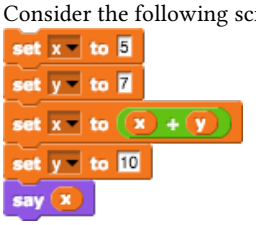
well on the questions that asked them to demonstrate basic content knowledge and were forthcoming with other evidence of CK. Eliciting aspects of pedagogical content knowledge proved to be more difficult and there was insufficient variation in the sample to analyze responses in terms of depth. However, teachers did identify common student errors that would be predicted by research. In the next phase of this project, we will be iterating on the assessment questions, expanding the item pool, and conducting additional phases of validity testing. Future validity testing will contain qualitative work, such as think-aloud studies with teachers completing the assessment and testing the items with a broader population of high school CS teachers. We also plan to expand the coding scheme to capture some indication of the depth of teacher knowledge being displayed. Another area for further exploration is how the different areas of teacher knowledge are developed. In our sample, pre-test to post-test change was most evident in responses to the explanations and representations questions, suggesting that knowledge development in this area may depend more heavily on practical experience than on direct instruction.

A APPENDIX

You have assigned your students homework that includes the following question.

Snap! version

Consider the following script.



What does the sprite say when the script is clicked?

Java version

Consider the following code segment.

```
int x = 5;
int y = 7;

x = x + y;
y = 10;

System.out.println(x);
```

What is the output?

What is the correct response? What incorrect responses do you expect to see from your students, and why do you think students will make those errors?

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1348866. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the National Science Foundation.

REFERENCES

[1] D. L. Ball, M. H. Thames, and G. Phelps. 2008. Content knowledge for teaching: What makes it special? *Journal of Teacher Education* 59, 5 (2008), 389-407. DOI: 10.1177/0022487108324554.

[2] Juliet A. Baxter. 1987. Teacher explanations in computer programming: A study of knowledge transformation. Stanford University, CA.

[3] Elena Bender, Peter Hubwieser, Niclas Schaper, Melanie Margaritis, Marc Berges, Laura Ohrndorf, Johannes Magenheim, and Sigrid Schubert. 2015. Towards a competency model for teaching computer science. *Peabody Journal of Education* (0161956X) 90, 4 (2015), 519–532.

[4] Malte Buchholz, Mara Saeli, and Carsten Schulte. 2013. PCK and reflection in computer science teacher education. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education (WiPSE '13)*. 8–16. https://doi.org/10.1145/2532748.2532752

[5] J. Gal-Ezer and C. Stephenson. 2010. Computer science teacher preparation is critical. *ACM Inroads* 1, 1 (2010), 61-66.

[6] S. Doukakis, A. Psaltidou, A. Stavraki, N. Adamopoulos, P. Tsiotakis, and S. Stergou. 2010. Measuring the technological pedagogical content knowledge (TPACK) of in-service teachers of computer science who teach algorithms and programming in upper secondary education. *Readings in Technology and Education: Proceedings of ICICTE,* 442-452.

[7] Nataša Grgurina, Erik Barendsen, Bert Zwaneveld, Klaas van Veen, and Idzard Stoker. 2014. Computational thinking skills in Dutch secondary education: Exploring pedagogical content knowledge. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research (Koli Calling '14)*, 173–174. https://doi.org/10.1145/2674683.2674704

[8] H. C. Hill, S. G. Schilling, and D. L. Ball. 2004. Developing measures of teachers' mathematics knowledge for teaching. *The Elementary School Journal* 105, 1 (2004), 11-30.

[9] H. C. Hill, B. Rowan, and D. L. Ball. 2005. Effects of teachers' mathematical knowledge for teaching on student achievement. *American Educational Research Journal* 42, 2 (2005), 371-406.

[10] A. Hubbard, Y. Kao, and D. Brown. 2016. Designing think-aloud interviews to elicit evidence of computer science pedagogical knowledge. Paper presented at the *Annual Meeting of the American Education Research Association.*

[11] Tami Lapidot. 2005. Computer science teachers' learning during their everyday work. Technion University, Israel.

[12] Neomi Liberman, Yifat Ben-David Kolikant, and Catriel Beeri. 2012. "Regressed Experts" as a new state in teachers' professional development: Lessons from computer science teachers' adjustments to substantial changes in the curriculum. *Computer Science Education* 22, 3 (2012), 257-283.

[13] John Loughran, Amanda Berry, and Pamela Mulhall. 2012. *Understanding and developing science teachers' pedagogical content knowledge, 2nd edition.* Sense Publishers. Retrieved from https://www.sensepublishers.com/media/1219-understanding-and-developing-science-teachers-pedagogical-content-knowledge.pdf

[14] P. Mishra and M. Koehler. 2006. Technological pedagogical content knowledge: A framework for teacher knowledge. *The Teachers College Record* 108, 6 (2006), 1017-1054.

[15] Laura Ohrndorf and Sigrid Schubert. 2013. Measurement of pedagogical content knowledge: students' knowledge and conceptions. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education (WiPSE '13)*, 104–107. https://doi.org/10.1145/2532748.2532758

[16] Laura Ohrndorf and Sigrid Schubert. 2014. Students' cognition: Outcomes from an initial study with student teachers. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE '14)*, 112–115. https://doi.org/10.1145/2670757.2670758

[17] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson. 2007. A survey of literature on the leaching of introductory programming. *SIGCSE Bull* 39, 4 (2007), 204-223.

[18] Mara Saeli, Jacob Perrenet, W. Jochems, and Bert Zwaneveld. 2010. Portraying the pedagogical content knowledge of programming—The technical report. Retrieved October 24, 2015 from https://www.tue.nl/fileadmin/content/universiteit/Over_de_universiteit/Eindhoven_School_of_Education/Onderzoek/Projecten_promovendi/Mara_Saeli_SPJZ_Technical_Report.pdf

[19] Mara Saeli, Jacob Perrenet, Wim MG Jochems, and Bert Zwaneveld. 2012. Pedagogical content knowledge in teaching material. *Journal of Educational Computing Research* 46, 3 (2012), 267–293.

[20] S. K. Selling, N. Garcia, and D. L. Ball. 2016. What does it take to develop assessments of mathematical knowledge for teaching?: Unpacking the mathematical work of teaching. *The Mathematics Enthusiast* 13, 1, 35-51.

[21] L. S. Shulman. 1986. Those who understand: Knowledge growth in teaching. *Educational Researcher* 15, 2 (2007), 4-14.

[22] J. Sorva. 2012. Visual programming simulation in introductory programming education (doctoral dissertation). Retrieved August 31, 2017 from Aaltodoc.