Measuring computer science pedagogical content knowledge: An exploratory analysis of teaching vignettes to measure teacher knowledge

Aman Yadav
Michigan State University
620 Farm Lane
East Lansing, MI 48824
ayadav@msu.edu

Marc Berges
Technical University of Munich
Arcisstr. 21
80333 Munich, Germany
berges@tum.de

Phil Sands
Michigan State University
620 Farm Lane
East Lansing, MI 48824
phil@msu.edu

Jon Good
Michigan State University
620 Farm Lane
East Lansing, MI 48824
goodjona@msu.edu

ABSTRACT

The initiatives to introduce Computer Science as a mandatory subject in K-12 in the U.S. (CSForAll), the U.K. (CAS), or Australia mean that thousands of new teachers will need to be trained both through inservice professional development and preservice teacher preparation. In order to examine the success of these efforts to train new computer science teachers requires computer science education researchers to evaluate the development of knowledge to teach computer science, i.e. pedagogical content knowledge. To date, we know little about how computer science pedagogical content knowledge looks like and how to assess it. This paper reports results from a qualitative analysis of computer science teachers' responses to teaching vignettes about students' understanding of programming constructs. The responses were evaluated using qualitative text analysis and commonalities are presented. In future research, the teachers' knowledge related to programming errors will be investigated on the basis of a survey developed from the answers of the presented study.

CCS Concepts

•Social and professional topics → Computing education;

Keywords

pedagogical content knowledge; qualitative content analysis; professional development

WiPSCE '16 October 13-15, 2016, Münster, Germany

DOI: http://dx.doi.org/10.1145/2978249.2978264

1. INTRODUCTION

A number of initiatives are currently underway to expand access to computer science (CS) to students in elementary and secondary classroom in the United States (CS for All), the U.K. (Computing At School) and Australia. In order for these efforts to be successful, thousands of teachers need to be trained to teach computer science. Teacher training at this scale is a tall order given the misconceptions about computer science education that continually permeate among teachers and administrators [6, 17]. In order to develop good computer science teaching, teachers not only require substantial understanding of computer science, but also a solid background in how to make the CS ideas comprehensible to students. These two areas of knowledge, referred to as "content knowledge" and "pedagogical knowledge", are significantly dependent on each other. Shulman extended the notion of knowledge of subject matter and knowledge about methods of teaching by introducing the idea of pedagogical content knowledge (PCK). He suggested that pedagogical content knowledge " goes beyond the knowledge of subject matter per se to the dimension of subject matter knowledge for teaching ... [and includes] the ways of representing and formulating the subject that make it comprehensible to others" [15, p. 9].

While the role of pedagogical content knowledge is well known in a number of subject areas including mathematics [8], and science [4], we know little about what this knowledge looks like in computer science. This limited understanding of computer science pedagogical content knowledge (CS-PCK) means that we know little about whether the professional development and teacher training efforts in computer science are being successful in developing knowledge teachers need to effectively teaching computer science. Hence, it is imperative to develop measures that assess computer science teachers' pedagogical content knowledge. To measure CS-PCK, we developed an open-ended questionnaire that asked teachers to make decisions in a realistic computer science pedagogical situations (i.e., teaching vignette). This paper presents a preliminary analysis of how computer science teachers addressed the given classroom situation related to

students' programming difficulties. Furthermore, we outline our next steps towards a measurement tool for computer-science teachers' pedagogical content knowledge related to programming errors.

2. RELATED WORK

Using qualitative analysis of teacher education curricula and interviews with computer-science teachers, a German research group developed a competence model for teaching computer science [1, 9]. Based on the analysis, the competence model included pedagogical content knowledge, teachers' beliefs, and motivational orientations with regard to teaching computer science. The pedagogical content knowledge competence factor consisted of two dimensions: *typical fields of pedagogical operations* (FPO) and *aspects of teaching and learning* (ATL). The first dimension, FPO, represented three categories withing the process of teaching and learning in the classroom, which included planning and design of learning situations (FPO 1), reacting on students' demands during instruction (FPO 2), and evaluation of teaching process (FPO3). The second dimension, ATL, included fifteen categories that related to subject matter/curriculum related issues, teaching methods and use of media, learner related issues, teacher related issues, and issues of the educational setting. The current analysis focuses on one of the FPO categories, 'reacting on student's demands during the instruction processes', and two ATL categories, 'subject specific teaching concepts' (i.e., Programming classes) and 'student cognition'. Researchers have also argued that conceptualization of pedagogical content knowledge includes teachers' ability to recognize and address student thinking [12, 7]. Hence, any measurement of CS-PCK requires teachers to respond to students' alternate conceptions, specifically addressing how they would assist the student(s).

Saeli et al. (2011) conducted a literature review on pedagogical content knowledge related to teaching programming to address the following questions - "what are the reasons to teach programming; what are the concepts we need to teach programming; what are the most common difficulties / misconceptions students encounter while learning to program; and how to teach this topic" [14]. Saeli and colleagues used these four questions to schematize and define PCK within the context of programming. In order to develop teaching vignettes and subsequent qualitative analysis, we focused on two of the questions about what learning difficulties students encounter when learning to program and how teachers could addressed them. The authors' investigation suggested that student difficulties in learning to program typically included, lack of understanding of notations (such as, syntax and semantics) and structures (such as, the schemas to reach small-scale goals like using a for loop vs. while loop). This informed the pedagogical situations we developed and the qualitative data analysis.

While the research on CS-PCK is limited, there is considerable prior work on student misconceptions related to programming. Clancy (2004) indexed a numerous examples of programming-related misconceptions [5]. The misconceptions typically arise from over- and under-generalization of knowledge transferred from other areas like English, Mathematics, or previous programming experiences as well as having confused computational model. Clancy argued that addressing these misconceptions should focus on learner's effort to link different knowledge and encouraging knowledge integration. Ragonis and Ben-Ari [13] listed 58 difficulties and misconceptions related to object oriented programming. In summary, while there is a lot of research on student misconceptions and difficulties, there is little evidence of teachers' knowledge on how to react on those misconceptions and errors. This study built upon this work related to student misconceptions to measure knowledge teachers need to address them.

3. METHODOLOGY

3.1 Participants

Twenty one computer science teachers from the United States of America participated in the study. Participants included 11 males and one female (remaining participants did not provide demographic information). In addition to computer science, teachers in this study also taught other subjects including mathematics (N=7), career and technical education (N=3), and engineering (N=4).

3.2 CS Pedagogical Content Knowledge Measure

An open-ended questionnaire was developed to assess teachers' computer science pedagogical content knowledge (CS-PCK). The questionnaire included 25 questions each of which presented a teaching vignette related to a computer science concept. Each vignette presented a computer science situation that teachers are likely to encounter working with the students and asked the participants to respond how they would address the problem in their own classroom. The use of vignettes has been found to be an effective way to get rich descriptions of how teachers respond in particular teaching context as well as measure their knowledge to teach content area (i.e., their pedagogical content knowledge) [4, 8, 16].

3.3 Procedures and Data Analysis

Fifty computer science teachers were invited to complete the computer science pedagogical content knowledge measure. Twenty one teachers completed the questionnaire, a 42% response rate, which has been found to be good [11].

The response to the open-ended questionnaire generated a rich qualitative data set, which was analyzed using content analysis. For that purpose, we combined each participant's responses in one document and used Mayring's [10] step-by-step model of summarizing content analysis. Step 1 of the process involved "determination of the units of analysis", which was the smallest text passage containing relevant content, e.g. a word, phrase or paragraph. During the next step all the information in the response not critical to the research question was eliminated and we began paraphrasing of the content-bearing text passages'. During step 3 the "envisaged level of abstraction" was determined and the paraphrases were generalized. The steps 4 to 6 involved transitioning of the generalized paraphrases into a category system.

Following this process, we divided each participant's response into text segments with each representing one thought of the participant. Afterwards, the text was coded deductively and categorized by paraphrasing and abstracting. Additionally, we categorized all the questions themselves into overarching problems because similar codes had varying meanings in the context of different questions.

Table 1: Most mentioned codes and corresponding sample responses for the overarching problems

Code	Sample Response
Overarching Problem 1 – Teacher addressing students' problem	
explore code step by step	I would walk the students through each operation of each problem step by step.
point student to the error (by question)	I would then press him to explain why his original solution is correct. The conversation would continue until Reggie understood.
revisit content	I would revisit the or statement and emphasize that the evaluation of the condition does not influence the possible outcomes
use real values for variables	We would sit down and give values to both x and y. Talk through the expressions and see if all statements are true. Remembering that or can be one or the other.
Overarching Problem 2 – Teacher helping student understanding CS concept	
ask question to let the student see their error	She should see that the code will not perform the way she is desiring, so I would guide her with questions to make modifications to her code until it works properly (probably starting with: So the value of x is going up by one, but what would you want it do do for a countdown? How would you get it to go down by 1 instead?)
explain underlying content again	I would explain that else is like the word otherwise or use the phrase "This or that" to explain if-else
explain what happened in the code	I would explain to Tim that x is assigned the value of 10, then a conditional is performed to see if x is equal to ten. If it is, we change the value of x by 1, if is not, we change the value of x by 5, but we don't do both. An if-else evaluates only one of the statements, not both.
write down/tell result of each step	I would help him understand by drawing a small table, label column 1 with a and column 2 with b. I would show him, in succession, a being assigned the value of 10, b being assigned the value of a (which is 10), and then a being assigned the value of 20 by crossing out the 10 and writing 20
Overarching Problem 3 – Teacher identifying students' problem	
change the input value and rethink	I would change the values around to determine what is giving her issues.
let the student step through the code	Next, explain how or demonstrate how the code with execute.
Overarching Problem 4 – Teacher assumption about students' reasoning	
"automatic" else	Ajay believes the condition in the 'if' block is true (he's correct), but probably misinterprets the last two blocks by thinking they are the else of the conditional and would be skipped.
automatic stop at output	Otherwise he stopped looking at the program once he saw an output and assumed that all programs only have one output.
ignoring if (wrong interpretation of condition)	His reasoning is that the second line of code sets x = 12 ignoring the if block of code below.

4. RESULTS

The 25 questions (i.e., teaching vignettes) from the questionnaire were categorized and generalized under four overarching problems. These four overarching problems were:

1. How would the teacher address the problem?
2. How would the teacher help a student understand a computer science concept?
3. How does the teacher identify students' problems?
4. What student reasoning does the teacher assume?

Each of the overarching problems were coded separately. The first overarching problem was how a teacher would address the student problem. Sixteen codes emerged from analyzing teachers' responses. Table 1 presents the four top codes that were revealed during the data analysis along with a sample response. These four codes represent how majority of the teachers in our study addressed the problem presented in the teaching vignette. For example, one teacher stated that he would "walk the student through each operation of each problem step by step" (i.e., explore code step by step).

The second overarching problem used to categorize the questions in the survey was how a teacher would help a student understand the computer science concept. For this problem, fifteen codes emerged from the teacher responses to the vignettes. Table 1 illustrates the four topmost codes from the analysis along with a sample response for each. For example, when teachers responded that they would explain the underlying concept again to the students, as highlighted one teacher who stated, "I would explain that else is like the word otherwise or use the phrase 'This or that' to explain if-else".

The third overarching problem was how teachers identified students' problems. In total, five codes emerged from the teacher responses and we have included the top two codes in Table 1.

The fourth overarching problem was how the teacher saw student's reasoning presented in the teaching vignette. In total, four codes emerged from the analysis of the teachers' responses. Table 1 shows the codes and corresponding sample response for the code with more than two occurrences.

5. DISCUSSION

The comparison of the overarching problems revealed a difference in depth of teachers' responses to the teaching vignettes. The coding of first and second problems exhibited that teachers were able to respond in several ways, the qualitative analysis of the third and fourth problems, on the other hand, had only a small variety in the answers. Additionally, the number of occurrences of a certain response varied across the four overarching problems as can be seen in the ratio of responses with more than two occurrences and those with less. A possible explanation is that the first two problems addressed observable issues of teaching while the

latter ones addressed hidden aspects of students' thinking during teaching.

Biggs's taxonomy [3] and the three levels of teaching experiences might provide an insight into these results. In the first level, the teacher focuses on what the student is. Here, "the purpose of teaching is to transmit information, usually by lecturing"[3]. The second level focuses on what the teacher does, e.g., teaching "is still conceived as a transmission process, but of concepts and understandings, not just of information". Finally, in the third level, the teacher focuses on what the student does and think about how a student understands concepts and principles in the way they ought to understand them. Results from our study demonstrate that most teacher responses were coded and could be categorized with the first two levels. Only few of the responses focused on what the student does. This suggests that teachers were adept at focusing on the teaching practices and were limited in examining student thinking presented in the teaching vignettes. We need to conduct further research on how to make students' thinking visible to computer science teachers, which has implications for the professional development of teachers.

6. CONCLUSION AND FUTURE RESEARCH

The investigation of the teachers' responses on the vignettes showed that there is a need to strengthen the efforts to educate teachers in addressing problems related to programming errors. More precisely, teachers seem to have difficulties if the students' thinking is latent. In the next steps of our research we have to find out if the qualitative levels suggested by Biggs [3] can be found through psychometric analysis. A quantitative analysis of the prospected data aiming for several levels of knowledge probably will lead to a validated measurement tool for the pedagogical content knowledge and the corresponding teaching competences.

7. ACKNOWLEDGMENTS

We would like to thank all the teachers who participated in this study. This work was funded by National Science Foundation (NSF) CE 21 program under grant 1502462. The opinions expressed are those of the authors and do not necessarily reflect those of NSF.

8. REFERENCES

[1] E. Bender, P. Hubwieser, N. Schaper, M. Margaritis, M. Berges, L. Ohrndorf, J. Magenheim, and S. Schubert. Towards a Competency Model for Teaching Computer Science. *Peabody Journal of Education*, 90(4):519–532, 2015.

[2] M. Berges, P. Hubwieser, J. Magenheim, E. Bender, K. Bröker, M. Margaritis-Kopecki, J. Neugebauer, N. Schaper, S. Schubert, and L. Ohrndorf. Developing a competency model for teaching computer science in schools. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, page 327, New York, USA, 2013. ACM.

[3] J. Biggs. What the Student Does: teaching for enhanced learning. *Higher Education Research & Development*, 18(1):57–75, 1999.

[4] D. Brovelli, K. Bölsterli, M. Rehm, and M. Wilhelm. Using Vignette Testing to Measure Student Science Teachers' Professional Competencies. *American Journal of Educational Research*, 2(7):555–558, 2014.

[5] M. Clancy. Misconceptions and Attitudes that Interfere with Learning to Program. In S. Fincher, editor, *Computer Science Education Research*, pages 85–100. Taylor & Francis, London, 2004.

[6] Google. Searching for computer science: Access and barriers in U.S. K-12 education. 2015.

[7] O. Hazzan, T. Lapidot, and N. Ragonis. Guide to teaching computer science: An activity-based approach. Springer, New York, 2014.

[8] H. C. Hill, D. L. Ball, and S. G. Schilling. Unpacking Pedagogical Content Knowledge: Conceptualizing and Measuring Teachers' Topic-Specific Knowledge of Students. *Journal for Research in Mathematics Education*, 39(4):372–400, 2008.

[9] M. Margaritis, J. Magenheim, P. Hubwieser, M. Berges, L. Ohrndorf, and S. Schubert. Development of a competency model for computer science teachers at secondary school level. In *IEEE Global Engineering Education Conference*, pages 211–220, Los Alamitos, 2015. IEEE Press.

[10] P. Mayring. *Qualitative content analysis: theoretical foundation, basic procedures and software solution.* 2014.

[11] D. D. Nulty. The adequacy of response rates to online and paper surveys: What can be done? *Assessment & Evaluation in Higher Education*, 33(3):301–314, 2008.

[12] L. Ohrndorf and S. Schubert. Students' cognition: outcomes from an initial study with student teachers. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, 112-115, New York, USA, 2014. ACM.

[13] N. Ragonis and M. Ben-Ari. A long-term investigation of the comprehension of OOP concepts by novices. *Computer science education*, 15(3):203–221, 2005.

[14] M. Saeli, J. Perrenet, Wim M. G. Jochems, and B. Zwaneveld. Teaching programming in secondary school: A pedagogical content knowledge perspective. *Informatics in Education*, 10(1):73–88, 2011.

[15] L. S. Shulman. Those Who Understand: Knowledge Growth in Teaching. 1986.

[16] A. Yadav. Video cases in teacher education: What role does task play in learning from video cases in two elementary education literacy methods courses? Michigan State University, East Lansing, MI, 2006.

[17] A. Yadav, C. Mayfield, N. Zhou, S. Hambrusch, and T. Korb. Computational thinking in elementary and secondary teacher education. volume 14, pages 1–16. 2014.