Teachers' Perspectives on Talk in the Programming Classroom : Language as a Mediator (Authors' pre-print version)

Sue Sentance
sue@raspberrypi.org
Raspberry Pi Foundation, Cambridge
Cambridge, UK

Jane Waite
j.l.waite@qmul.ac.uk
Queen Mary University of London
London, UK

ABSTRACT

Motivation. In education, classroom talk is a vital aspect of a lesson, and programming education is no exception. While the role of language and dialogue has been researched in depth in other school subjects, there has been less research in the programming context. Sociocultural theory highlights the importance of language as a mediator for learning, alongside other tools.
Objectives. Drawing on sociocultural theory and models of dialogic education, the purpose of the study was to investigate the ways in which programming teachers use classroom talk to support learning, and to propose a model to frame our understanding of this element of programming lessons.
Method. The qualitative study used phenomenological methodology to investigate and interpret teachers' 'lived experiences' of classroom talk. Interviews were conducted with 20 primary and secondary computing teachers about the content and effect of classroom talk in programming lessons. The context of the study was PRIMM, a lesson structure which highlights the importance of talk around a shared programming artefact.
Results. Analysis of data revealed four main themes: how talk occurs in the classroom setting, how questioning is used to facilitate talk, how students are encouraged to explain, and why teachers feel it is important for students to use correct vocabulary.
Discussion. Building on research into models of dialogue in education and our findings we suggest a model to frame talk in the programming classroom. We discuss the contribution of PRIMM to our understanding of talk in programming lessons. More research is needed to validate the proposed model and to investigate the impact of classroom talk on learning outcomes in programming.

CCS CONCEPTS

• **Social and professional topics** → **K-12 education**.

KEYWORDS

dialogue, K-12 computing education, PRIMM, programming, sociocultural theory, vocabulary

ACM Reference Format:
Sue Sentance and Jane Waite. 2021. Teachers' Perspectives on Talk in the Programming Classroom : Language as a Mediator (Authors' pre-print version). In *Proceedings of the 17th ACM Conference on International Computing Education Research (ICER 2021), August 16–19, 2021, Virtual Event, USA*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3446871.3469751

1 INTRODUCTION

Many countries are moving to a curriculum that includes more emphasis on algorithms and programming [24, 26, 30, 74], and this is reflected in an increase in programming education for K-12 research. Research falls broadly into several camps, for example, tools and environments to support programming, instructional approaches, resource development, diversity, assessment and teacher education. There is little research, however, on specific aspects of programming lessons such as classroom talk, and how they can impact learning [74]. This contrasts with other disciplines, where classroom talk has been studied in more depth (see for example, [17, 43, 53, 54, 62]). In programming lessons, classroom talk will include: the dialogue that students have with each other and their teacher about their programs, the use of technical language, the types of questions teachers and students ask, and how concepts are explained and exemplified. In so much as these impact on students' progress in their acquisition of knowledge and skills, it is important that we understand the relative role of these elements of a lesson.

We owe much to sociocultural theory in enabling us to understand how language can support learning. Through the theoretical lens of the soviet psychologist Vygotsky, language can be seen as a central form of mediation that enables thinking and internalisation of concepts to take place [76]. As Jerome Bruner says of Vygotsky:

> *"His basic view was that conceptual learning was a collaborative enterprise involving an adult who enters into dialogue with the child in a fashion that provides the child with hints and props that allow him to bring a new climb, guiding the child in next steps even before the child is capable of recognising their significance."* [12, p.852]

There have been calls to take a more sociocultural approach within computer science education and programming [48, 72] and particularly regarding the role of language in teaching computer science [18], but otherwise this is a largely unexplored area of research. Just as in science teaching, where different ways of talking in class about concepts lead to internalisation of the necessary concepts [41], in computer science education we need to use talk to help students to internalise the difficult concepts they face in programming.

This paper describes an exploratory research study looking at classroom talk in the context of programming education, from teachers' perspectives. The study is situated in the context of PRIMM, a structure for programming lessons [66]. PRIMM particularly focuses on classroom discussion, specific questioning about code, and asking students to talk to each other about code, based around a sample program that has been carefully constructed to prompt discussion and learning [67].

2 THEORETICAL FRAMEWORK

In this section we set out our theoretical perspective, in terms of sociocultural theory, its influence on dialogic models and how it relates to programming education.

2.1 Sociocultural theory

Social constructivism, in particular the work of the Soviet psychologist Vygotsky, can frame our understanding of novice programmers and their learning. Vygotsky proposed that higher mental processes were functions of mediated activity and that there were three major classes of mediators: material tools, psychological tools (sometimes called signs and symbols), and other human beings [37]. Mediation is a key focus of Vygotsky's work [82], and includes both the nature of the tools adopted and valued by society as well as the appropriation of tools and how they are integrated into cognitive activity during the processes of an individual's development [68]. By understanding mediation and the process of appropriation better in computing education, we may be more able to support novice learners better, particularly with respect to programming.

There are a group of theories under the banner of sociocultural theory [49] but here we focus on Vygotsky's sociocultural theory (SCT). In the context of school learning, Vygotsky states that a child's development within their Zone of Proximal Development (ZPD) involves social interaction, dialogue, and mediated activity between learners and with their teachers [77]. The term proximal (nearby) indicates that the assistance provided goes slightly beyond the learner's current competence, complementing and building on their existing abilities. This means identifying what students can and cannot do, through tasks, then facilitating learning through tasks carefully situated within the ZPD which enable the student to carry out more complex cognitive tasks than they would be able to do on their own, with the support of a 'more knowledgeable other' (MKO). Students finally move to a stage where the student can work independently and reach their own learning goals [77].

One of the psychological tools described by Vygotsky is language, a central form of mediation that enables thinking and internalisation of concepts to take place [78]. As a communicative tool, language facilitates individuals' social interaction; as a psychological tool, language enables individuals to internalize the knowledge and skills in social interaction [16]. Language provides the means for scientific ideas to be talked through between people on the social plane, before internalisation [41].

2.2 Computing education through a sociocultural lens

There have been calls to take a more sociocultural approach within computer science education and programming [48, 72], and sociocultural approaches have strongly influenced a range of recent computer science education work [6, 13, 63]. Other recent research has specifically referenced or used Vygotsky's ZPD [5, 35, 36], indicating that the influence of sociocultural theory (SCT) is increasing its prominence in computer science education.

Sentance et al. [67] highlight three key educational principles from SCT that can guide the teaching of programming, and which are embedded in the PRIMM framework:

(1) **The role of the 'more knowledgeable other' (MKO) in ZPD**. Students need teachers, as MKOs, to show them (model) how to solve a problem. Students working together can be paired so that one peer is the MKO. Resources, materials and lesson structure are all mediating activity in Vygotsky's terms, but should be designed to be within the learners' ZPD. This requires a detailed understanding of progression and which concepts are easier or harder for students [83].
(2) **Learning moves from the social plane to the psychological plane**. According to Vygotsky, the process of mastering a semiotic tool typically begins on the 'social plane', which means the first stages of acquaintance typically involve social interaction and negotiation between experts and novices or among novices [81]. By participating in this social interaction interpretations are first proposed and worked out and, therefore, become available cognitively to individuals, into what Vygotsky calls the 'psychological plane' [79]. Using starter or example programs and teaching strategies to support the reading of code involves program code existing on the social plane initially, before being understood internally by the student. Programming tasks should be carefully chosen so they are within the ZPD of the student. Gradually more complex programming tasks involving independent problem solving and creativity will be possible by students as understanding becomes internalised.
(3) **Mediation through language**. When learning to understand how programs work, students should be encouraged to discuss with each other through a social construction of knowledge. This can be through pair programming, or collaborative tasks such as talking about segments of program code to identify their function. Teaching should facilitate focused discussion around programming constructs and concepts and their implementation as code.

PRIMM has been designed as a structure for programming lessons drawing on these principles. A PRIMM lesson has four components: the shared artefact of the program; the structure defined by the PRIMM acronym (predict, run, investigate, modify and make); the lesson resources; and structured talk. All four are mediators in the Vygotskian sense. In addition, the MKO, whether this be a teacher or a more knowledgeable peer, has a role to play in supporting these mediating activities, and the lesson resources and structured classroom talk should be within the ZPD of the students [67]. These elements provide the model underlying PRIMM, of which classroom talk is one significant part. This paper sets out to expand on the language/talk element of this model.

2.3 Dialogic models

Across a number of disciplines, including mathematics and science education, the role of language has been explored through various models of how dialogue works. We next introduce four influential models.

Firstly, Mercer and colleagues, drawing on SCT, developed the idea of *exploratory talk* [53], in which partners engage critically but constructively with each other's ideas. Exploratory talk is contrasted with disputational talk and cumulative talk [55]. Compared

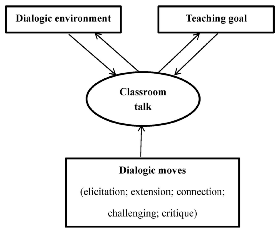

Figure 1: Framework for dialogue in education [16, p. 11]

with the other two types, exploratory talk involves knowledge being made more publicly accountable, with reasoning being more visible in the talk.

Another approach is Nystrand et al.'s *dialogically organised instruction* [59] which sets out three teacher discourse moves frequently used by dialogic teachers to organise instruction coherently: uptake (incorporating student ideas into subsequent questions of other students), authentic questioning (used to explore views not test knowledge), and high-level evaluation (where the teacher incorporates the response into elaborative comments).

A third approach is *dialogic teaching* [1]. Dialogic teaching focuses on the use of discussion and dialogue in the classroom, influenced by the work of Vygotsky and Bruner. Alexander proposes that teachers should promote dialogue that is collective, reciprocal, supportive, cumulative and purposeful [2], thus emphasizing both the classroom as a dialogic environment and the teaching goal of classroom talk.

Finally, a fourth approach to dialogue centres on *accountability* [56], whereby students should engage in respectful and grounded discussion, should listen to each other, and understand wait time and turn-taking. Talk should be accountable to the community (respectful), accountable to knowledge (domain-specific), and accountable to accepted standards of reasoning (supporting explanation and self-correction) [56].

Much empirical work has been conducted to validate these different theoretical frameworks for dialogue in education (see Section 3). Recently, Cui and Teo compared the four models to consider similarities and differences [16]. Accountability to the learning community embraces ideas similar to the principles of collectivity and reciprocity in dialogic teaching, while accountability to knowledge echoes the principles of cumulation and purposefulness. Thus they assert that Accountable Talk [56] has much in common with the five principles of dialogic teaching [1], and that Thinking Together [31, 54] has much in common with dialogically organized instruction [59] and dialogic teaching, because all emphasize the importance of cultivating a dialogic classroom culture by highlighting the role of ground rules.

Cui and Teo [16] developed a framework with which to consider the field of dialogue in the classroom as a whole, as shown in Figure 1. We can interpret the proposed framework with reference to a programming lesson. *Teaching goals* may be syntax, programming vocabulary, concepts, constructs, etc. The *dialogic environment* is the classroom context facilitated by the teacher from moment to moment. *Dialogue moves* may be types of questions, explanations, challenges, and corrections. In this paper we use the insights from Cui and Teo's analysis to develop our understanding of language and talk in the programming classroom in particular. By doing this we develop a new model for talk in the programming classroom by focusing on language as one of the core ingredients of a programming lesson situated in a sociocultural approach to learning.

3 RELATED WORK

3.1 Dialogue and talk

3.1.1 Talk in computing education. In computing education, most of the literature relating to language and communication as a vehicle for learning centres on pair programming and peer instruction [74], both privileging classroom talk and purposeful dialogue. Research has shown that peer instruction positively impacts learning outcomes [60, 87]. Pair programming has been shown to improve program quality and confidence [7, 52], although in the school context it may depend on the way that the collaborative work is instantiated [45].

Tsan et al. [73] conducted a study of six pairs of fifth grade students (ages 9–11) in the United States, on an elective programming course. They examined students' dialogue considering how they balance dialogue, turn-taking and control when learning to program. Their study revealed specific dialogue strategies used by students such as 'Let me help you' or 'Make Suggestion' [73]. Another study which looked at interaction mechanisms in computing students' talk identified collaborative problem-solving, conversations expressing excitement, and more social conversations [33].

3.1.2 Dialogue in other discipline-based education research. Dialogic models were described as part of our theoretical framework. Many empirical studies have been conducted to evaluate these, with some examples given here.

To measure the impact of exploratory talk a series of research projects were conducted under the banner of *Thinking Together*. The research involved interventions that gave both teachers and students new skills in using language for reasoning. In mathematics, this was shown to enable them to use language more effectively as a tool for working on maths problems together. One study involving 406 children and 14 teachers over a two-year period found that improving the quality of children's use of language for reasoning together improved their learning and understanding of mathematics and that the teacher is an important model and guide for pupils' use of language for reasoning [54]. Another study examined video data from 72 demographically diverse classrooms, with pupils aged 10-11 studying mathematics, literacy and science. All classes included productive teacher–student dialogue and this was measured against outcomes including standardised UK student assessments in literacy and mathematics. The study found that three aspects of teacher-student dialogue strongly predicted pupil performance: elaboration (building on contributions), querying (challenging a contribution) and participation (where students engage with each other's ideas) [31]. In another recent large-scale randomised controlled trial examining the impact of dialogue teaching [3], it was found that focusing on meaningful dialogue had a positive impact on primary school children's attainment, engagement and overall learning. This study involved over 4000 9-10 year old pupils split between control and intervention groups in over 70 schools in England.

In these studies a key element of the interventions was professional development for teachers in supporting quality dialogue in the classroom, in order that children could develop the linguistic tools needed to develop 'productive talk' [17]. As Dawes outlines in the context of science:

> *"The sort of language tools used in science, are, for example, questioning, explaining, putting things clearly, repeating or rephrasing, predicting, reasoning, evaluating, deciding. But the child's linguistic toolkit may not be so extensive or develop. In addition, children may be completely unaware of the potential power of talk with their peers."*[17, p. 684]

Children asked to work collaboratively or talk together with their peers about a problem are often not sure what they are expected to do [54] and are not able to bring to the task the necessary skills. In the Thinking Together project studies described above, teachers in the intervention groups were given training that helped them learn strategies for facilitating productive talk. They were also provided with lesson plans in the subject areas (mathematics or science) that included the development of talk skills such as critical questioning, sharing information and negotiating a decision [54].

3.1.3 *Questioning.*

There is a substantial and decades-long literature on the topic of questioning that is beyond the scope of this paper including, amongst other topics, how teachers ask open and closed questions [29], the value of wait time before answers [14], and demographic differences relating to who teachers ask which questions to [85]. One area of questioning that has led to some debate is the use of Initiation-Response-Feedback (IRF) style of questions [70] to elicit answers from students where the answers to the questions are already known. These types of questions have been criticised for inhibiting classroom talk and the development of ideas [17, 84]. However, IRF questions have their place, although in the framework of the dialogic models we introduced in Section 2, the focus is more likely to be on open, exploratory questions.

In the studies described above, a central part of the analysis was the questions used by the teachers, in particular the degree to which teachers used 'why' questions.

3.2 Vocabulary

Another aspect of language is the use of technical discipline-specific language that helps students understand their subject.

3.2.1 *Computing vocabulary.*

Diethelm and Goschler highlight the lack of attention to computing-specific vocabulary [18] and consider that specific items of computing vocabulary may be ambiguous or have different meanings in everyday life from their scientific meaning. They suggest a need for a *meta-discourse* around language such that pupils in school can learn to distinguish between everyday and scientific meanings of terms, and that teachers should be more deliberate about vocabulary [19]. There is clearly scope for more detailed investigation into how young learners acquire and use the technical vocabulary in programming.

3.2.2 *Vocabulary in other disciplines.*

There has been a considerable focus on the language of school science, and how students can acquire and use it in the classroom, highly influenced by a much-cited text by Lemke [43]. Lemke purports that students do not just talk about science, but that they *do* science through the medium of language, and that the acquisition of vocabulary and semantics is essential to understanding. The language of any subject is not just its special vocabulary, but also the semantic relations constructed between the words as we use them [43]; thus understanding of science and language go hand in hand [17, 50].

Vocabulary is also important in mathematics education: teachers need support in understanding where elements of vocabulary cause difficulty for students [62]. Leung [44] suggests that in mathematics education there are three related processes involved in vocabulary learning: learning formal and semantic features of words in different contexts, learning the concepts associated with the words, and incremental meaning-making as understanding develops. We can relate this to programming education: using the example of the term 'iteration' the school student may at first be introduced to the word, then the concept through many examples, and then develop their own understanding incrementally as they use the construct in practical programming examples.

3.3 PRIMM for programming lessons

PRIMM [66] is an approach to structuring programming lesson. Lessons (or sequences of lessons) are structured using the following activities:

- **Predict** what code will do
- **Run** the code to test predictions
- **Investigate** the structure of code
- **Modify** the code to add functionality
- **Make** a new program using the same/modified structures.

A central element of the approach is that a piece of code is in the 'social plane' initially as a shared artefact for discussion and comprehension. The PRIMM lesson involves discussion between teacher-class, teacher-student and student-student about a piece of code, while unpacking its structure and function. The 'investigate' element of the PRIMM structure relates to the Block Model framework and how it frames program comprehension [34, 64]. The Block Model highlights aspects of the program such as function of the whole, structure of a block, execution of an atomic item within the code, etc. and a PRIMM lesson includes questions and activities that are designed to engage students in discussion about all aspects of how the code works and is structured. The Modify-Make stages of PRIMM are also similar to the Use-Modify-Create model [42], although the earlier stages are quite different. In addition, PRIMM focuses on reading and understanding code. Code comprehension is already readily accepted in programming education literature, including the importance of reading code and being able to trace what it does before writing new code[46, 47]. Research has demonstrated that novices require a 50% tracing code accuracy before they can independently write code with confidence [47, 75].

PRIMM has been used in primary and secondary school classrooms, particularly in England, since 2018. Several studies have investigated its impact, including a mixed-methods study conducted in 2018 involving around 500 students aged 11 to 14 [67]. In this study, a quasi-experimental design was used to investigate the impact of a series of PRIMM-structured lessons on learner outcomes. Teachers delivered programming lessons using the PRIMM approach for 8 to 12 weeks. Data was collected via a combination of a baseline test, a post test to compare control and experimental groups, and teacher interviews. The results showed a statistically significant difference in the score between the control and experimental groups for all students favouring the experimental group. Qualitative results highlighted that teachers particularly value the collaborative approach taken in PRIMM, and the structure given to the lessons [66]. Another study involving PRIMM illustrated its use for games programming in higher education [40].

3.4 Summary and research question

We have seen that research in mathematics and science education seems to demonstrate that if teachers and students are provided with guidance and training in effective dialogue in the classroom, there will be improvement in learning outcomes. Research into discourse in school computing lessons is an emerging field, including exploring talk in pair programming and vocabulary use more generally.

PRIMM, as an approach in which programming is taught through an apparently discourse rich series of activities, draws on the idea of language as a mediator for learning, central to SCT. To explore classroom talk a context needs to be selected. Given that there is much still we need to understand about the teaching and learning of programming the focus of this study is to explore the role of classroom talk in the PRIMM context. This leads us to the following research question: *In what ways do teachers develop classroom talk to support the learning of programming?*

4 THE STUDY

A study was conducted to investigate teachers' use of classroom talk in programming lessons. Drawing on the literature described above, the purpose was to explore the nature of classroom talk in the programming classroom and develop a model representing teachers' experiences of classroom talk and programming that could be evaluated and potentially used in the classroom and for future research.

4.1 Methodology

Qualitative research was chosen for this study to provide the richness and detail needed to answer the research question [38]. To ensure an interpretive approach to data collection and analysis, we adopted hermeneutic phenomenological methodology. Phenomenological research seeks to reveal and describe 'lived experiences' and to achieve a deeper understanding of the meaning of experience, generating an in-depth and comprehensive description of the phenomenon [57], and can be descriptive or interpretive; a hermeneutic (interpretive) phenomenological approach involves interpreting and making meaning out of participants' lived experiences. Hermeneutic phenomenology emerged from the work of hermeneutic philosophers, including Heidegger, Gadamer, and Ricoeur, who argue for our embeddedness in the world of language and social relationships, and the inescapable historicity of all understanding [20].

When using hermeneutic (interpretive) phenomenology as a methodology, it is not just the analysis and interpretation of data that draws on the phenomenological principles. The recruitment of participants, sampling, data collection and interview structure should also reflect an interpretive approach [22]. In terms of the data analysis, reflexivity can help interpret the meanings discovered [71]. Reflexivity involves intensive scrutiny about how something is known and/or understood [32] and involves researchers being conscious of and reflective of how their questions, methods and subject position might impact the data [71].

With the goal of a true examination of shared experience, qualitative data analysis remains as close to the data as possible and focuses on what participants say and how they say it [61]. Themes can be viewed as written interpretations of lived experience [71]. Identifying themes is an iterative and recursive process and starts with the researcher's engagement with the data during data collection and the early stages of reading and re-reading the data.

4.2 Participants

Hermeneutic phenomenological research necessitates a homogenous group of individuals; participants should demonstrate experience of the same phenomenon [15] but be diverse enough to enhance possibilities of "rich and unique stories" [39, p. 29]. For this reason, purposeful sampling was employed, given our responsibility to select participants who had an important and meaningful experience of the phenomenon [86]. We focused our purposive sampling on teachers who had used PRIMM. Participants were recruited through email and social media.

4.3 Data collection

An interview schedule was designed as shown in Table 2, which focused on open questions around the teachers' experiences of interactions and talk in the programming classroom. Interviews were held online and scheduled over two weeks in 2020; each interview lasted 30-45 minutes. Interviews were audio-recorded before being professionally transcribed. Participants were given the opportunity to check their transcripts for accuracy and add any further reflections, and the data was then carefully anonymised.

4.4 Data analysis

Reflexive thematic analysis is a theoretically-flexible data analysis approach which involves searching across a data set to find repeated patterns of meaning [8], and is an appropriate guide for data analysis for phenomenological researchers. It involves prolonged engagement with the data, including reflexive journalling by the researcher [58], using detailed notes and memos during the entire process. Reflexive thematic analysis procedures centre on organic and recursive coding processes, and the importance of deep reflection on and engagement with data [9]. The process we used for data analysis is shown in Figure 2, primarily aligned to that described in [58], and incorporating multiple stages of interpretation [39].

The qualitative data analysis (QDA) software NVivo was used to work with the data, and both authors were involved in the analysis of the data. The first author conducted the interviews, wrote detailed memos and carried out detailed coding of each interview,

Table 1: Experience and characteristics of study participants

Phase	Teaching Experience	12 years or more		4 - 11 years		Up to 3 years	
	PRIMM experience / Gender	<= 1 year	>1 year	<= 1 year	>1 year	<= 1 year	>1 year
Primary	Male		Teacher L				
			Teacher N				
			Teacher Q				
	Female						
Secondary	Male	Teacher B	Teacher G	Teacher D	Teacher A	Teacher I	
		Teacher E	Teacher H	Teacher K	Teacher P	Teacher O	
		Teacher M	Teacher R	Teacher T			
		Teacher S					
	Female				Teacher C		
			Teacher J		Teacher F		

Interview questions	
General background	Experience of teaching Experience of teaching programming Experience of teaching PRIMM
Talk-related questions	Types of talk that take place in programming lessons? Prompt if needed: experience of any of the following - Teacher talks to whole group (instructions, explanations) - Teacher asks questions (individual, group) - Students/pupils ask questions (of teacher, of each other) - Students/pupils talk to each other Talk differences in lessons structured/not structured with PRIMM Difficulties students/pupils have in talking about programming Role in fostering discussion amongst your students
Open comment	Any other comments

Table 2: Themes of interview questions

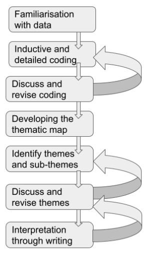
Familiarisation with data
Inductive and detailed coding
Discuss and revise coding
Developing the thematic map
Identify themes and sub-themes
Discuss and revise themes
Interpretation through writing

Figure 2: The data analysis process

capturing and labelling the comments that represented lived experiences [22] of the use of language and talk in the context of programming lessons, and other opinions and reflections of the teachers. The second author followed through these processes, becoming familiar with the data through reading and re-reading, and carried out independent coding.

The thematic map [10] was developed by both authors as a composite summary of all the interviews and was a precursor to the development of specific themes that captured the contributions being made through the interviews. Developing the thematic map involved reviewing, combining and recombining descriptive codes, and was discursive and iterative. Interpretation and discussion is the key aspect of a phenomenological analysis: consensus and agreement are not reported in this approach [51].

After summarising the interviews, the thematic map was categorised into domains of interest (general topics) and those domains that were pertinent to the research question were identified as primary themes for further synthesis. A theme is understood as a set of experiences that many teachers referred to in the interviews, whereas a domain of interest refers to the general topic that teachers talked to, guided in part by the interview structure. A jointly-conducted iterative process continued to develop the sub-themes to further guide the analysis and discussion. A sub-theme enables the analysis to be more finely grained, but not all teachers may refer to each sub-theme. The process used was to *"move in and out of the detail iteratively"* [22, p.10] through re-reading and discussion amongst the researchers.

4.5 Validity and credibility

Interpretive research is not driven by an absolute definition of reality in the way that more positivist and quantitative approaches to qualitative research are [65]. The validity of interpretive research can be seen as the extent to which the constructions of the researcher are grounded in the constructions of those being studied [21]. To address validity of the data analysis process we used Guba and Lincoln's notion of trustworthiness [28], as expanded on more recently by Shenton [69]. Trustworthiness is now widely accepted as a way of ensuring validity and reliability in qualitative data analysis [61]. The four elements of trustworthiness are credibility, transferability, dependability and confirmability, which align respectively to internal validity, external validity, reliability and objectivity in quantitative research [27]. We attended to credibility following Shenton's criteria, by using established research methods, by ensuring we were completely familiar with the context of the participating teachers, developing a relationship with the participants that would support integrity and honesty in their reports, by focusing on their lived experiences rather than opinions, and by using field notes memos within the QDA software as a reflective commentary [69]. Transferability is achieved in part by thick descriptions, and contextual information about the participants. Following Shenton [69], we have provided some information about the participants in terms of their background and have detailed the process of data collection. However, more detail of each particpant's context is prohibited due to the number of teachers involved in the study. For dependability and confirmability, we have reported the decisions made in our study in as much detail as space provides, and used the iterative interpretive process to examine our own biases and ensure the analysis reflects the lived experiences of each participant.

4.6 Ethical considerations

Ethical procedures outlined in [11] were followed; participants gave consent to the use of their data for specific purposes and full information was given. After transcription, participants were able to check their interview transcripts.

5 FINDINGS

The thematic map was developed as part of the data analysis process (see Figure 2) and is shown in Figure 3. From this we can see that teachers discussed a range of different aspects of interaction and talk in their classrooms.

5.1 Identification of themes

After the conclusion of the process described in Section 4.4, nine themes were identified across the data as a whole:

- Student difficulties and differences when learning to program
- Using writing and annotating to learn to program
- How programming talk occurs in the classroom
- How questions are used in the classroom
- Why students' verbal explanations about the code are important and how they are encouraged
- Why students' use of correct programming vocabulary matters and how it is encouraged
- How PRIMM effects talk or about PRIMM in general

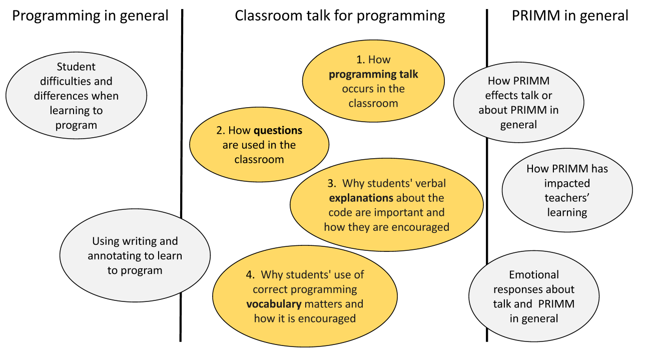
Programming in general
Classroom talk for programming
PRIMM in general
Student difficulties and differences when learning to program
Using writing and annotating to learn to program
1. How programming talk occurs in the classroom
2. How questions are used in the classroom
3. Why students' verbal explanations about the code are important and how they are encouraged
4. Why students' use of correct programming vocabulary matters and how it is encouraged
How PRIMM effects talk or about PRIMM in general
How PRIMM has impacted teachers' learning
Emotional responses about talk and PRIMM in general

Figure 3: Thematic Map

- How PRIMM had impacted teachers' learning
- Emotional responses about talk and PRIMM in general

These themes were then categorised within three overarching domains of interest: programming in general, PRIMM in general, and classroom talk for programming. This categorisation is shown in Figure 3 via the vertical parallel lines. As our domain of interest is classroom talk for programming the four themes fully within this scope were selected for analysis:

- Theme 1: How programming talk occurs in the classroom
- Theme 2: How questions are used in the classroom
- Theme 3: Why students' verbal explanations about the code are important and how they are encouraged
- Theme 4: Why students' use of correct programming vocabulary matters and how it is encouraged

From these themes we see that encouragement around language was very pertinent to teachers' experiences. Sub-themes were identified and are shown, along with relevant cases in Table 3. Each theme will be discussed in turn, and for each sub-theme, we identify common attributes and outliers where applicable.

5.2 Theme 1: How programming talk occurs in the classroom

In describing how talk was encouraged in the classroom, three sub-themes emerged:

- What the example program contributes
- What the teacher contributes
- What student-student interaction contributes

These sub-themes often overlapped, for example, when teachers referenced their dialogue with students about the program. The significance of the example or starter program (an aspect of a PRIMM lesson) was vital for many teachers.

Teachers repeatedly mentioned students referring to the shared example program when they were talking in computing classes. They commented on how having tasks around a program, rather than starting with a blank screen, resulted in greater focus and high-quality talk. One teacher experienced that this would keep students 'on task':

> " *[the talk] is more focused because they're already into the task and it avoids that situation of the student going, I just can't do this, I'm going to talk to my mate about*

Table 3: Themes, sub-themes, and occurrences

Theme	Sub-theme	Teachers
Theme 1: How programming talk occurs in the classroom setting	a) What the example program contributes	All except H
	b) What the teacher contributes	B-E, G-T
	c) What student-student interaction contributes	B-G,I,J,L,N-P,R-T
Theme 2: How questions are used in the classroom	a) How questions relate to the example program and activities	B,C,F,L,N,P-S
	b) Questions generated by the teachers	M,O,P,R,S,T
	c) Questions generated by students	C,F,G,I,M-O,R
Theme 3: Why students' verbal explanations about the program code are important and how encouraged	a) Explanations centred around program code	A-J,Q,S,T
	b) Teachers' explanations	A,B,F,I,L,R,T
	c) Students' explanations	A-G,I,J,P,Q,R,T
Theme 4: Why students' use of correct programming vocabulary matters and how encouraged	a) Vocabulary to pass examinations	C,E-H,P,R,S,T
	b) Vocabulary to support accurate mental model	G-J,L,N,O,P,S,T
	c) Vocabulary-specific activities	B,J,L,O,P,S,T
	d) Teacher modelling vocabulary use	A,E,F,H-N,Q-T
	e) Students using programming vocabulary	A-G,I,K,L,M,N,Q-T

the footy and stuff because there's no point." (Teacher M) (Theme 1 sub-theme a)

Another teacher, fairly new to teaching, talked about how having example code led to discussion that had more depth:

> *"... the level of discussion I'm having is much deeper. Where kids don't seem to be as concerned with... there might still be some syntactical errors, which is fine, that they're dealing with, but it's not just about how to get something basic running; it is a little bit that. They're going a lot deeper."* (Teacher O) (Theme 1 sub-theme a)

Teachers often mentioned their role in classroom talk, including leading whole class discussions, instigating one to one or small group discussions and responding to pupils' requests to talk about the code or an activity. A change in the type of talk was highlighted, such as becoming more of a *facilitator* by using structured activities. This was exemplified by a teacher in the context of a point he was making about the *Run* phase of PRIMM:

> *"So there's definitely more student talk than teacher talk at that point, and then I've become more of a facilitator, and I'm just going round with different students and just pointing bits out to them where they've got a bit confused and misconceptions."* (Teacher N) (Theme 1 sub-theme b)

In talking about lessons, teachers described the protocols, patterns, and atmosphere of talk and how they fostered this to promote talk :

> *"I'm asking the question to the whole room, and I'm doing no hands up, and I'm doing cold call. I'm doing more think-pair-share and partner talk."* (Teacher B) (Theme 1 sub-theme b)

Teachers highlighted the importance of encouraging peer discussions. Teachers talked about peer talk creating opportunities for students to learn by talking to each other, to help each other solve problems, and how talk revealed student's understanding, making it visible to teachers.

With all themes there are commonalities across the data set with occasional 'unique voices' [25, p.51]. For example, a teacher also mentioned his view that for some students talk was a distraction:

> *"Some children are much more verbal than others, and so they think out loud whereas some children ... would just get distracted by that, and they just need to almost tinker with the code, for want of a better word, to investigate it, but it needs to be guided."* (Teacher N) (Theme 1 sub-theme c)

Phenomenological research involves zooming in and out of the detail [22] and although space does not permit an in-depth reporting of our analysis for each teacher, we can present some extracts from one teacher in particular across this theme, to examine how his wider experience impacted on how he used language in his teaching. To do this, we consider Teacher R, a highly experienced male teacher. He trained as a science teacher, then also taught ICT (Information and Communications Technology); he has learned to teach computing since it was introduced into the school curriculum in 2014. Teacher R describes his emphasis on whole-class explanation:

> *"I need to explain everything about what's going on. So, yes, I would probably explain quite a lot. That's my teaching style. Coming from a science background, I would explain things anyway.... The pupils would therefore talk less"*(Teacher R) (Theme 1 sub-theme b)

There is some repetition of the word 'explain' in this extract: it is mentioned three times. The teacher is keen to emphasise that doing this is important to him and that it is his own style of teaching, and he expects the pupils to talk less at this stage of the lesson. He leans on his experience of science to justify his teacher-led approach, as if this is not common in computing lessons (from the teachers we interviewed it varies). However he reflects that he subsequently moves on to programming activities where he expects students to talk to each other, to help each other out, and talk about their work. Here he lists a number of questions that may take place at this stage in the lesson:

> *"Then when they're going with their programming, what they would do is they would probably talk to each other more if they come across problems, or they'll talk to me, and I would be asking them questions like 'Why does it not work?', 'What can you spot?', 'How is this different to the example that you've got?' … 'Why does that example work and yours doesn't?' "* (Teacher R) (Theme 1 sub-theme c)

The teacher is providing the opportunity for the students to talk to each other about their code, and is available to have one-to-one dialogues about the code. However the questioning seems to be focused on syntax, with the notion of a 'problem' being solved when a difference in syntax between a student's code and an example is identified. This teacher then asks students to write explanations of what their code does in their exercise books. The teacher does use talk in the lesson, but also values writing to consolidate knowledge. Teacher R then reflects on listening to students' talk at different points in the lesson:

> *"… yes, you listen. So, 'what isn't working? What is working?' … if you're talking through a specific problem with a particular student, then obviously listening is important, because otherwise you can't pick up on what they're thinking."* (Teacher R) (Theme 1 sub-theme b)

Teacher R differs from several of the other teachers in his focus on teacher-led explanations. Here he reflects on the fact that the researcher has raised the topic of listening, which he says later is *'making him think'*, so the interaction between the researcher and teacher has stimulated a line of thought that may not necessarily have otherwise arisen in his account of his experience. Many of the teachers mentioned through the interview that they were reflecting on their practice during the interview.

5.3 Theme 2: How questions are used in the classroom

Teachers mentioned questions as an essential aspect of talk in their classrooms. Comments in this theme clustered into three sub-themes:

- How questions related to the example program and activities
- Questions generated by the teacher
- Questions generated by students

Again, these three sub-themes often overlapped, for example, the teacher asking pupils about the example program or the teacher reflecting on the way students asked each other questions about the activity at hand.

Teachers talked about the the shared example program that had been created specifically to teach about the lesson topic and how this framed and led questions, enabling them to ask whole-class, small group and individual student questions on the same topic. For example, Teacher B described how he asked the whole class a question about the shared code, requiring students to talk to their partner. He would then take responses from the whole class, followed by probing questions for specific learners, such as asking what a particular command would do in the example or asking learners to identify an instance of a programming construct.

While some questions the teachers described were intended to prompt technical vocabulary use or predictions of what example code might do when run, others were used to prompt deeper thinking, asking why the program did something or asking students to compare or evaluate code. Here a primary teacher uses the starter code as a focus for questioning:

> *"… and then I'd say, well, why do you think that's happening? Where in the code do you think that it's doing this? Then if they couldn't see it, we'd maybe step through it together. "* (Teacher L) (Theme 2 sub-theme a)

One teacher explained that he sometimes started with a closed question and moved to open questions that stretched learners:

> *"What line number do you first see a variable on? And it's the type of thing that anyone can just have a guess at, even if they don't know. And then we go to more open-ended ones."* (Teacher P) (Theme 2 sub-theme b)

Other than talking about open and closed questions, teachers did not categorise their questions. One very experienced teacher, M, discussed the difficulty he had with asking good questions.

A secondary teacher was surprised to hear students re-using her questions as they talked to each other:

> *"I could almost hear myself in their voices … I would hear - But why does that work? Why is yours better than mine? How can I make mine look yours and still make it work? Yours looks more efficient, explain to me why it's more efficient."* (Teacher C) (Theme 2 sub-theme c)

Several teachers commented that students used 'deeper' questions due to the example program. Several teachers described that the PRIMM teaching approach elicited more advanced questioning between teacher and pupil:

> *"… students ask questions, they are also asking at a deeper level too. They seem to jump straight into a more advanced topic right away. There are still a few people who get confused by some of the syntax and some of the basics, but there's less of that. "*(Teacher O) (Theme 2 sub-theme c)

5.4 Theme 3: Why students' verbal explanations about the code are important and how they are encouraged

Using explanations is not unexpected in terms of teachers' description of talk in the programming classroom. The three sub-themes identified within the area of explanations were:

- Explanations centred around the program code
- Teachers' explanations
- Students' explanations

There were overlaps between the three sub-themes, for example where the code supported a student explanation. To exemplify this, several teachers talked about students' explanations about code facilitating learning, generating a 'ah ha moment':

> *"Most of the time they're articulating what's going on or what they think is going on. Then they would go, ah, I see, yes, right. And they would figure it out for themselves."* (Teacher D) (Theme 3 sub-theme a)

Teachers reported that their experience of teaching programming had led them to routinely encourage student explanations to foster understanding. Student explanations could be focused on the example code or their own programs, for example:

> *"I start to approach them and suddenly they look at their screen again as if they're ready to explain it to me and then they're like, ah, I know what it is . . . It's getting that cognitive discourse going in your head. I think that's really important."* (Teacher E) (Theme 3 sub-theme a)

Some teachers reported that their explaining to the whole class was central to their teaching style, as we saw in Theme 1. This contrasts with other teachers who reported that PRIMM activities around sample code meant that their need to explain to the whole class was much less. One of the primary teachers found this to be quite a stark change in his behaviour:

> *"I very much find that I'm talking for the first five minutes, maybe ten, and then it's very much over to the pupils working individually. So most of the rest of the lesson would be them talking unless we come across something that we'd need to talk about together. So I go around the room at that point. I'm asking questions."* (Teacher L) (Theme 3 sub-theme b)

As well as whole class explanations, teachers mentioned asking specific students to explain their programs. Sometimes these explanations required students to say what each line of the program would do, dry-running the code. Several teachers mentioned that students, even their older, more experienced students, found walking through and explaining their programs hard. However, they found that students explaining to each other what they did understand was important:

> *"So they can code it, but they're not particularly good at explaining and that challenges them a bit. It's quite nice to see them talking at the middle layer of students, as it were, talking to those, going, we haven't written much there, and I thought about this."* (Teacher I) (Theme 3 sub-theme c)

Teachers also mentioned students found summarising the algorithm or flow of control of a program challenging. One teacher recounted how he gave students a small whiteboard and he asked them to draw their explanation:

> *"I say, just doodle what's in your head. Just dump it down there. Do arrows like a mind map or mind bubble, whatever you want to call them, and then just draw the links between them."* (Teacher A) (Theme 3 sub-theme c)

Teachers highlighted that not all students had the confidence or inclination to explain their understanding. Teacher P reflected that he had not yet built up confidence in his class to an extent where they would explain their programs to each other independently in pairs, but it was happening in the whole class. Another teacher's experience was that once his students have their code working, he noticed a tendency for students to explain less verbally, instead switching to writing explanations.

5.5 Theme 4: Why students' use of correct programming vocabulary matters and how it is encouraged

The final theme relates to the technical vocabulary that is associated with programming. This might be conceptual terms such as assignment, iteration, or selection, or words relating to syntax, such as *if*, *for*, or the names of blocks in a block-based language. Throughout our discussion with teachers on classroom talk, they mentioned using the 'correct' or 'right' programming vocabulary and terms. These comments were grouped into the following five sub-themes:

- Vocabulary to pass examinations
- Vocabulary to support accurate mental model
- Vocabulary-specific activities
- Teachers modelling vocabulary use
- Students using programming vocabulary

The 'correct' vocabulary for teaching programming to students at different stages of learning to program is not the focus of this study and requires further investigation. However, teachers believe in the existence of 'correct' vocabulary and that it should be known and used by students.

Some teachers stated that students needed to know the 'correct' vocabulary to pass exams. Teachers also mentioned that students needed to know and use the 'right' terms. Teachers mentioned the need for a common language for learning and sharing understanding:

> *". . . that's about getting them to understand that these technical terms are important because obviously it helps them understand, and it helps them explain themselves to other people."* (Teacher S) (Theme 4 sub-theme a)

Building a mental model of concepts was mentioned by teachers:

> *" . . . Which is why we try and give them a language because the language helps them to express themselves better when they're talking about it. And also it helps them I think to have a mental model of what that is, if you give it a name. "* (Teacher J) (Theme 4 sub-theme b)

Here the teacher explicitly highlights that the terminology enables the learners to express themselves more clearly. This does contrast with some other teachers who were focused on correctness as a learning goal, or for examinations. To help learners acquire an understanding of the 'right' terms, teachers talked about using vocabulary specific tasks, or 'keyword activities', for example:

> *""I liked what Dylan was saying in his answer and the way he used keywords. Alisa, can you tell me which keywords you think I really liked?""* (Teacher B) (Theme 4 sub-theme c)

Another teacher, P, had created a whole bank of keyword games including identifying keywords and writing programs that use specific keywords. Other teachers mentioned using classroom displays to support students learning vocabulary.

Several teachers said they modelled programming terminology to reinforce what the terms meant and to encourage students to use it in class. Several teachers described banning words such as *thing* and *it*. One teacher described his perspective using a classroom example:

> *"I'm quite a stickler for the correct terminology . . . because when I hear something like, 'oh, yes, and then something happens' . . . I will frequently make that into a plenary activity. I'm like, right, so [Student Name] has said this word. What did she mean? Oh, she meant iteration. Great. So, I'm not going to stop them in their tracks to go, you need to speak like this, but I will positively reinforce using the technical terminology in that lesson."* (Teacher K) (Theme 4 sub-theme d)

The same teacher, representative of others, reflected that his experience is that students need time to gradually learn to use the terms correctly:

> *"It is just slow and gentle encouragement because they are learning a new language. They are learning a whole group of new syntax. It is gentle encouragement that's needed, but I do feel they do need to use the correct terminology eventually . . . And when it comes to the exams, we need them to be as clear as possible in the larger writing questions, and that clarity is gained through having a larger vocabulary."* (Teacher K) (Theme 4 sub-theme a)

As well as using the 'correct' vocabulary in examinations and with their teachers to develop and explain their understanding, teachers described the terminology that students used to talk to each other:

> *" . . . at the very start you wouldn't hear any of that correct vocabulary . It's like a foreign language, really, for them. Because they've never really come across it before, and it's not necessarily how you talk either."* (Teacher D) (Theme 4 sub-theme e)

Not all teachers said that they always required the right vocabulary, with one teacher describing his reaction to students using non-standard vocabulary:

> *"Even though I will overhear them [using incorrect vocabulary], I wouldn't stop them. I'd rather they just explain it in their own words rather than get too forced into the right terminology."* (Teacher I) (Theme 4 sub-theme e)

5.6 Summary of findings

The themes identified and illustrated above reveal insights into how computing teachers talk about talk. The first theme, around the tools that teachers use to encourage talk in the classroom highlighted shared program code as a contributor. The second theme highlighted types of questions and the way they were used to encourage not only dialogue and discussion, but at a deeper or more advanced level. The third theme we found highlighted the explanations that students use, and the fourth theme focused on technical vocabulary, whether precision of terminology was important to teachers, and why.

For each of the first three themes, the three sub-themes relate to the language focus, the teacher's experience and the student's experience. At the point of analysis we could have developed themes along these lines, but our decision to highlight themes on the basis of types of language: talk in general, questions, explanations and vocabulary/terminology made it easier to reflect the different experiences teachers reported. The fourth theme on vocabulary actually links all the themes, as teachers were very keen to express how important 'correct' programming terminology was, although they had different reasons for doing so, as indicated by the sub-themes.

6 DISCUSSION

The interviews provide a rich story of teachers' experience in the programming classroom with pupils of various ages. In this section we review and interpret the findings with relation to the research question '*In what ways do teachers develop classroom talk to support the learning of programming?*' through the development of a model to represent the themes we have identified in the data. We then abstract some of the generic elements of this data-generated model to produce a potentially useful framework to view the role of classroom talk in programming lessons.

6.1 Developing a model

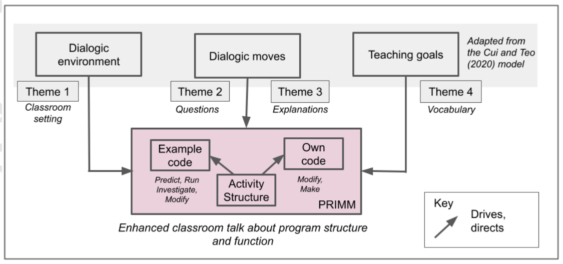

Figure 4: Talk in the programming classroom - data driven model

Figure 4 shows a model which incorporates Cui and Teo's dialogic education framework as shown in Figure 1 [16]', and its relationship to the themes we have been exploring. Cui and Teo identified dialogic environment, dialogic moves and teaching goals as inputs to classroom talk. Given that the current study explored language and talk in the context of PRIMM lessons, we have also represented elements of PRIMM as a context in the model. We use this model to further interpret the themes highlighted in Section 5.

6.2 Instantiating the model

In Theme 1, **How programming talk occurs in the classroom setting**, teachers painted a picture of the roles and norms set up by teachers to facilitate talk. We mapped this theme to the dialogic environment. The environment, or atmosphere of a setting, are those established expectations, relationships and behaviour patterns that encourage and support talk helping it flourish [16]. Teachers also highlight the time and effort needed to create routines which establish patterns of talk, in order to create a dialogic environment.

The effort needed to develop a dialogic classroom culture is a known issue with time and processes needed to establish or change routines and expectations and the underpinning teacher-student, student-student relationships [1, 16].

PRIMM afforded teachers and students a predictable activity structure that directed talk about the example code during the Predict, Run, Investigate and Modify phases. The activity and example code acted as a mediator for learning [81]. To represent the role of these mediating tools in the dialogic classroom, we have added PRIMM as a context and the activity structure and example code to our emerging data-driven model as shown Figure 4.

We have mapped Themes 2 and 3 to *dialogic moves* in the proposed model. For Cui and Teo, *dialogic moves* represent the convergence of the discourse models they synthesised with respect to "classroom talk strategies" [16, p.12] including:

- eliciting a contribution such as through authentic questioning
- extending dialogue by asking learners to explain through elaboration or substantiation
- connecting links between participants and their contributions
- challenging participants to clarify and deepen thinking
- critiquing through critical evaluation of each others' contributions

In Theme 2, **How questions are used in the programming classroom**, teachers reflect on how they use questions to elicit dialogue. In Theme 3, **Why students' verbal explanations about the code are important and how they are encouraged**, teachers describe other *moves* that they make to encourage students to articulate their understanding of a program by explaining how it works. Our study clearly shows that the teachers reporting their classroom experiences are using talk strategies to support students' learning of programming concepts, constructs, syntactic elements and problem solving skills. We saw some evidence of elaboration, where teachers or peers invite or provide elaboration on a previous contribution. Elaboration is one of the aspects of teacher-student dialogue found to strongly predict performance in assessments [31], as was querying, which we saw in our data, where teachers reported that their students were asking increasingly in-depth questions of each other about their code.

However, some teachers are more restricted to question types which have been criticised as closing down talk, such as Initiation-Response-Feedback (IRF) style of questions [17, 84], where the teacher already knows the answer, for example, asking where a variable is in the code.

Compared to the dialogue-trained teachers participating in the *Thinking Together* research and other studies in mathematics and science education focusing on classroom dialogue [31, 54], the teachers in this study do not seem consciously familiar with the breadth of dialogic techniques that they could use. This aligns to previous work in mathematics:

> "*There are good reasons to expect that children studying maths would benefit from teacher guidance in two main ways. First and most obviously, they need to be helped to gain relevant knowledge of mathematical operations, procedures, terms and concepts. Teachers commonly expect to provide this kind of guidance. Secondly, they need to be helped to learn how to use language to work effectively together: to jointly enquire, reason, and consider information, to share and negotiate their ideas, and to make joint decisions. This kind of guidance is not usually offered.*" [54, p. 410]

Some teachers in our study clearly have not reflected on classroom talk before, although they comment on how the conversation with the researcher is causing them to reflect on language more.

There is evidence (for example, [3, 54]) that when teachers are able to model and guide students in dialogic strategies that children use dialogue more effectively, reason better, and have better learning outcomes. Despite the teachers' lack of experience in dialogic techniques, we do see that the teachers see themselves as a model and a guide [54]. Several teachers talked about themselves as 'facilitators'.

Studies as part of the dialogic education research described in Section 3 included training teachers and students to use different ways of deepening a dialogue to enable more reasoning. This would be a useful next step for programming education, as using dialogue to reason about a program could be a transformational tool for the classroom.

We mapped Theme 4 to *Teaching Goals* where the goals are used during lesson planning and as the lesson unfolds to drive the dialogue to be purposeful for a particular subject, according to the learners needs and interests [16]. In Theme 4, **Why it is important for students to use correct programming vocabulary and how this is encouraged**, it was clear that teachers were insistent on 'correct' vocabulary use as a teaching goal. This included both explicitly teaching terms through to modelling them.

What we do not know from the teachers' reflections is what those terms mean to the students. Other research has discussed the difference between everyday meanings of words and technical terms [17, 19] and that this can cause confusion. Some teachers referenced literacy difficulties that their students had and how the acquisition of programming terminology could help that. Beyond the programming aspect of computing, it seems clear that children's development of conceptual language in computing is an important area that could unlock some useful insights into our understanding of the learning process, as researched in other subjects [50].

6.3 The shared artefact as a mediator

Across all four themes emerging from the interviews was the contribution of the shared programs. PRIMM promotes the use of starter code as a focus for a series of activities, such as predicting what code will do and a range of other code comprehension exercises. Students are encouraged to develop understanding of how a piece of code works before starting to modify it and eventually build their own. The teaching goals for the lesson are focused around a program that exemplifies the underpinning concepts being introduced, and a discussion that includes explaining out loud how programs might work and supporting students to develop a 'language' to do this. There is already evidence that teachers find this approach useful [66].

PRIMM provides a particular context where the lesson follows a structure with a range of activities and exercises around example and student's own code, and Figure 4 shows how the PRIMM context encourages classroom talk around a shared artefact. Emerging from theme 1 was the idea that the shared program code gave a context for discussion that enhanced both the quantity of articulation and the depth or quality of the discussion about the code. This was also apparent in the second theme, where the shared example code from the Predict and Investigate stages of PRIMM was described by teachers as the source of many different types of questions and explanations.

We have seen that sociocultural theory emphasises the importance of tools. Language is one tool, as is the teacher (MKO), but also resources and examples provide a mediator for learning [81]. Furthermore the shared example code is not owned by the student, but in the 'social plane' [80], so there it can be a conduit for talk without being personalised. Therefore, in the context of a programming classroom, the shared activity and shared code are social tools which are mediating the learning and the discourse.

6.4 Generalising and evaluating the model

We have noted that teachers report that students do not find it easy to explain how a program works, or to use a range of linguistic tools to verbalise their reasoning. Programming teachers are not generally trained to facilitate productive dialogue in our experience. Furthermore many teachers in our study reported delivering whole-class explanations rather than focusing on the ways in which learners could improve their own explanations, despite the fact that student-student dialogue is a focus of PRIMM, in which context this study was set. In any programming lesson, not just those using the PRIMM structure, the research on dialogue in mathematics and science suggests that it would be useful to focus professional development on dialogue moves around the explanations of programs and how teachers can facilitate their students developing their linguistic skills in this area.

All subjects have a degree of 'disciplinary literacy' in that there are technical terms that enable students to create a shared understanding when discussing the subject with each other; being able to effectively use these can support the development of a mental model of particular concepts. Again, this is a language focus that is not specific to a PRIMM-style lesson but there is much for us to understand around the introduction of technical vocabulary and the associated learning goals. Several teachers in our study were focused primarily on the fact that some terminology was 'correct' and for that reason it should be encouraged, rather than associating the use of programming terminology with a growing understanding of programming itself.

Having developed a model which represents the interpreted data aligned to the theoretical framework and context in which the study is situated, we are thus proposing a generic model for broader contexts, as presented in Figure 5. The generic model highlights two areas where we feel more research is needed, which relate to our understanding of the dialogic environment and appropriate dialogic moves for programming lessons. The PRIMM context is now removed, as any programming lesson with a dialogic focus could draw on this model.

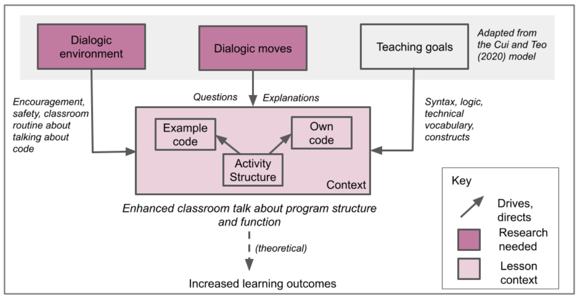
Dialogic environment
Dialogic moves
Teaching goals
Adapted from the Cui and Teo (2020) model
Encouragement, safety, classroom routine about talking about code
Questions
Explanations
Syntax, logic, technical vocabulary, constructs
Example code
Own code
Activity Structure
Context
Enhanced classroom talk about program structure and function
(theoretical)
Increased learning outcomes
Key
Drives, directs
Research needed
Lesson context

Figure 5: Talk in the programming classroom - generic model

This generic model includes different elements of a programming lesson and the ways in which dialogue can enhance classroom talk. Further studies would clearly be needed to validate the proposed model. One approach would be to replicate or adapt some of the studies conducted in other disciplines around dialogic teaching and education. Another angle to the research could be to focus on how the use of programming terminology links to more productive talk and evidence of learning.

6.5 Reflections and limitations

As researchers utilising a hermeneutic phenomenological methodology within this exploratory study, which requires a considerable amount of reflexivity and self-reflection, it is valuable to reflect on our own experience and learning. In contrast to approaches to qualitative research that require a degree of quantitative analysis around coding, we were careful to pay most attention to teacher reflections that really reflected their lived experience, rather than opinions and views. We were conscious of our own experience and what we brought to the interpretation whilst conducting a level of 'bracketing' [39] to ensure that it did not interfere with the analysis.

The limitations of this approach in this study was that the number of participants (20) was quite high for an interpretive phenomenological study, generating a lot of data which we could have explored in much depth. For example, we did not draw on some of the teaching history and personal experiences of the teachers that we recorded in analysing the extracts as much as we could have. In presenting the themes, we sacrificed some depth to demonstrate the commonality of the themes across the data, because as researchers who also share classroom experience, we felt these elements of the teachers' experiences were important to draw out.

Hermeneutic phenomenological methodology is not common in computing education research (although it is in other fields such as general education and nursing research [4, 23]), and it would be interesting to see other studies in computing education discuss this trade-off while utilising this methodology.

7 CONCLUSION AND FURTHER WORK

In this paper we have described a qualitative exploratory study into teachers' perspectives around the language and talk they use in the programming classroom. We used hermeneutic phenomenology as the methodology and drew on sociocultural theory and dialogic models to support our interpretation of teachers' reported lived experiences. 20 teachers who teach programming in primary or secondary schools using the PRIMM approach were interviewed for the study.

The findings in this paper suggest that teachers are very aware of the need for key programming terminology to give pupils a language to talk about their programs. Teachers describe the importance of the starter or example code (as used in PRIMM lessons) in providing a focus for dialogue, questions and explanations. Teachers describe different ways in which they encourage talk in their classroom, and see themselves as a guide and a model [54]. Student-student interaction is seen as important to learning but some teachers reflect that they do not encourage this as much as they think they should or could.

Through synthesis of our interpretation of teacher experiences of discourse in their programming lessons with dialogic theoretical frameworks we have developed a generic model to frame the way context specific shared artefacts such as starter/ example code, students' own code and activity structure can provide a focus for different types of classroom talk. We suggest two specific areas of further work. Firstly, to evaluate the proposed generic model by developing and evaluating a dialogic techniques for the programming classroom intervention. Such an intervention should build upon dialogic research in other subjects, such as in mathematics and science education. Secondly, to explore what the 'correct' vocabulary is for the learning of programming, how it might be effectively taught and what the impact is of the learning of programming 'vocabulary' on the development of conceptual understanding.

REFERENCES

[1] Robin Alexander. 2006. *Towards dialogic teaching: Rethinking classroom talk.* Dialogos, Cambridge.

[2] Robin Alexander. 2013. *Essays on pedagogy.* Routledge.

[3] Robin Alexander. 2018. Developing dialogic teaching: Genesis, process, trial. *Research Papers in Education* 33, 5 (2018), 561–598.

[4] Fidaa S Abu Ali, Lubna Abushaikha, et al. 2019. Hermeneutics in nursing studies: an integrative review. *Open Journal of Nursing* 9, 02 (2019), 137.

[5] Ashok R Basawapatna, Alexander Repenning, Kyu Han Koh, and Hilarie Nickerson. 2013. The zones of proximal flow: guiding students through a space of computational thinking skills and challenges. In *Proceedings of the ninth annual international ACM conference on International computing education research.* ACM, 67–74.

[6] Jens Bennedsen and Ole Eriksen. 2006. Categorizing pedagogical patterns by teaching activities and pedagogical values. *Computer Science Education* 16, 2 (2006), 157–172.

[7] Grant Braught, L Martin Eby, and Tim Wahls. 2008. The effects of pair-programming on individual programming skill. *ACM SIGCSE Bulletin* 40, 1 (2008), 200–204.

[8] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[9] Virginia Braun and Victoria Clarke. 2019. Reflecting on reflexive thematic analysis. *Qualitative Research in Sport, Exercise and Health* 11, 4 (2019), 589–597.

[10] Virginia Braun and Victoria Clarke. 2020. One size fits all? What counts as quality practice in (reflexive) thematic analysis? *Qualitative research in psychology* (2020), 1–25.

[11] British Educational Research Association (BERA). 2018. Ethical Guidelines for Educational Research, Fourth Edition. https://www.bera.ac.uk/researchers-resources/publications/ethical-guidelines-for-educational-research-2018

[12] Jerome Bruner. 1982. The language of education. *Social Research* 49, 4 (1982), 835–853.

[13] Åsa Cajander, Mats Daniels, and Roger McDermott. 2012. On valuing peers: theories of learning and intercultural competence. *Computer Science Education* 22, 4 (2012), 319–342.

[14] Kathleen Cotton. 1988. Classroom questioning. *School improvement research series* 5 (1988), 1–22.

[15] John W Creswell. 2007. *Qualitative inquiry and research design: Choosing among five approaches.* SAGE Publications, Inc; Second Edition.

[16] Ruiguo Cui and Peter Teo. 2020. Dialogic education for classroom teaching: a critical review. *Language and Education* 0, 0 (Oct. 2020), 1–17. https://doi.org/10.1080/09500782.2020.1837859

[17] Lyn Dawes. 2004. Talk and learning in classroom science. *International journal of science education* 26, 6 (2004), 677–695.

[18] Ira Diethelm and Juliana Goschler. 2015. Questions on spoken language and terminology for teaching computer science. In *Proceedings of the 2015 ACM conference on innovation and technology in computer science education, ITICSE '15.* ACM, 21–26.

[19] Ira Diethelm, Juliana Goschler, and Timo Lampe. 2018. Language and Computing. In *Computer Science Education: Perspectives on Teaching and Learning in School,* Sue Sentance, Erik Barendsen, and Carsten Schulte (Eds.). 207–219.

[20] Linda Finlay. 2012. Debating phenomenological methods. In *Hermeneutic phenomenology in education.* Brill Sense, 15–37.

[21] Uwe Flick. 1998. *An introduction to qualitative research.* Sage Publications Limited.

[22] Julie Frechette, Vasiliki Bitzas, Monique Aubry, Kelley Kilpatrick, and Mélanie Lavoie-Tremblay. 2020. Capturing lived experience: Methodological considerations for interpretive phenomenological inquiry. *International Journal of Qualitative Methods* 19 (2020), 1609406920907254.

[23] Norm Friesen, Carina Henriksson, and Tone Saevi. 2012. *Hermeneutic phenomenology in education: Method and practice.* Vol. 4. Springer Science & Business Media.

[24] Alexandra Funke, Katharina Geldreich, and Peter Hubwieser. 2017. Analysis of scratch projects of an introductory programming course for primary school students. In *2017 IEEE global engineering education conference (EDUCON).* IEEE, 1229–1236.

[25] Thomas Groenewald. 2004. A phenomenological research design illustrated. *International journal of qualitative methods* 3, 1 (2004), 42–55.

[26] Shuchi Grover, Nicholas Jackiw, and Patrik Lundh. 2019. Concepts before coding: non-programming interactives to advance learning of introductory programming concepts in middle school. *Computer Science Education* 29, 2-3 (2019), 106–135.

[27] Egon G Guba. 1981. Criteria for assessing the trustworthiness of naturalistic inquiries. *Ectj* 29, 2 (1981), 75–91.

[28] Egon G Guba and Yvonna S Lincoln. 1982. Epistemological and methodological bases of naturalistic inquiry. *ECTJ* 30, 4 (1982), 233–252.

[29] C Lynn Hancock. 1995. Implementing the assessment standards for school mathematics: Enhancing mathematics learning with open-ended questions. *The Mathematics Teacher* 88, 6 (1995), 496–499.

[30] Fredrik Heintz, Linda Mannila, Lars-Åke Nordén, Peter Parnes, and Björn Regnell. 2017. Introducing programming and digital competence in Swedish K-9 education. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives.* Springer, 117–128.

[31] Christine Howe, Sara Hennessy, Neil Mercer, Maria Vrikki, and Lisa Wheatley. 2019. Teacher–Student Dialogue During Classroom Teaching: Does It Really Impact on Student Outcomes? *Journal of the Learning Sciences* 28, 4-5 (Oct. 2019), 462–512. https://doi.org/10.1080/10508406.2019.1573730

[32] Kerry E Howell. 2012. *An introduction to the philosophy of methodology.* Sage.

[33] Maya Israel, Quentin M. Wherfel, Saadeddine Shehab, Oliver Melvin, and Todd Lash. 2017. Describing Elementary Students' Interactions in K-5 Puzzle-based Computer Science Environments using the Collaborative Computing Observation Instrument (C-COI). In *Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17).* Association for Computing Machinery, New York, NY, USA, 110–117. https://doi.org/10.1145/3105726.3106167 00014.

[34] Cruz Izu, Carsten Schulte, Ashish Aggarwal, Quintin Cutts, Rodrigo Duran, Mirela Gutica, Birte Heinemann, Eileen Kraemer, Violetta Lonati, Claudio Mirolo, et al. 2019. Fostering program comprehension in novice programmers-learning activities and learning trajectories. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education.* 27–52.

[35] Nadia Kasto. 2016. *Learning to Program: The development of knowledge in Novice Programmers.* Ph.D. Dissertation. Auckland University of Technology.

[36] Donna Kotsopoulos, Lisa Floyd, Steven Khan, Immaculate Kizito Namukasa, Sowmya Somanath, Jessica Weber, and Chris Yiu. 2017. A pedagogical framework for computational thinking. *Digital Experiences in Mathematics Education* 3, 2 (2017), 154–171.

[37] Alex Kozulin and Barbara Z Presseisen. 1995. Mediated learning experience and psychological tools: Vygotsky's and Feuerstein's perspectives in a study of student learning. *Educational psychologist* 30, 2 (1995), 67–75.

[38] Udo Kuckartz. 2014. *Qualitative text analysis: A guide to methods, practice and using software.* Sage.

[39] Susann M Laverty. 2003. Hermeneutic phenomenology and phenomenology: A comparison of historical and methodological considerations. *International journal of qualitative methods* 2, 3 (2003), 21–35.

[40] Robert Law. 2020. A Pedagogical Approach to Teaching Game Programming: Using the PRIMM Approach. In *European Conference on Games Based Learning.* Academic Conferences International Limited, 816–XVI.

[41] John Leach and Phil Scott. 2003. Individual and sociocultural views of learning in science education. *Science & Education* 12, 1 (2003), 91–113.

[42] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. Computational thinking for youth in practice. *ACM Inroads* 2, 1 (2011), 32.

[43] Jay L Lemke. 1990. *Talking science: Language, Learning, and Values.* Ablex Publishing, Norwood, NJ.

[44] Constant Leung. 2005. Mathematical vocabulary: Fixers of knowledge or points of exploration? *Language and Education* 19, 2 (2005), 126–134.

[45] Colleen M. Lewis. 2011. Is pair programming more effective than other forms of collaboration for young students? *Computer Science Education* 21, 2 (June 2011), 105–134. https://doi.org/10.1080/08993408.2011.579805

[46] Raymond Lister, Elizabeth S Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, et al. 2004. A multi-national study of reading and tracing skills in novice programmers. In *ACM SIGCSE Bulletin*, Vol. 36. ACM, 119–150.

[47] Raymond Lister, Colin Fidge, and Donna Teague. 2009. Further Evidence of a Relationship Between Explaining, Tracing and Writing Skills in Introductory Programming. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education* (Paris, France) *(ITiCSE '09)*. ACM, New York, NY, USA, 161–165.

[48] Philip Machanick. 2007. A social construction approach to computer science education. *Computer Science Education* 17, 1 (March 2007), 1–20. https://doi.org/10.1080/08993400600971067

[49] Lauren Margulieux, Brian Dorn, and Kristin Searle. 2019. Learning sciences for computing education. In *The Cambridge Handbook of Computing Education Research*, Sally Fincher and Anthony V. Robins (Eds.). Cambridge University Press, 208–230.

[50] Karl Maton and Yaegan Doran. 2021. *Constellating science: How relations among ideas help build knowledge.* Routledge (in press).

[51] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and inter-rater reliability in qualitative research: Norms and guidelines for CSCW and HCI practice. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–23.

[52] Charlie McDowell, Linda Werner, Heather E. Bullock, and Julian Fernald. 2006. Pair programming improves student retention, confidence, and program quality. *Commun. ACM* 49, 8 (2006), 90–95. https://doi.org/10.1145/1145287.1145293

[53] Neil Mercer. 1995. *The Guided Construction of Knowledge: Talk amongst teachers and learners.* Multilingual matters.

[54] Neil Mercer and Claire Sams. 2006. Teaching Children How to Use Language to Solve Maths Problems. *Language and Education* 20, 6 (Nov. 2006), 507–528. https://doi.org/10.2167/le678.0 00416 Publisher: Routledge _eprint: https://doi.org/10.2167/le678.0.

[55] Neil Mercer and Rupert Wegerif. 1999. Is 'exploratory talk'productive talk. *Learning with computers: Analyzing productive interaction* (1999), 79–101.

[56] Sarah Michaels, Catherine O'Connor, and Lauren B Resnick. 2008. Deliberative discourse idealized and realized: Accountable talk in the classroom and in civic life. *Studies in philosophy and education* 27, 4 (2008), 283–297.

[57] Clark Moustakas. 1994. *Phenomenological research methods.* Sage publications.

[58] Lorelli S. Nowell, Jill M. Norris, Deborah E. White, and Nancy J. Moules. 2017. Thematic Analysis: Striving to Meet the Trustworthiness Criteria. *International Journal of Qualitative Methods* 16, 1 (Dec. 2017), 1609406917733847. https://doi.org/10.1177/1609406917733847 02780 Publisher: SAGE Publications Inc.

[59] Martin Nystrand, Lawrence L. Wu, Adam Gamoran, Susie Zeiser, and Daniel A. Long. 2003. Questions in Time: Investigating the Structure and Dynamics of Unfolding Classroom Discourse. *Discourse Processes* 35, 2 (2003), 135–198. https://doi.org/10.1207/S15326950DP3502_3

[60] Leo Porter, Cynthia Bailey Lee, Beth Simon, and Daniel Zingaro. 2011. Peer instruction: do students really learn from peer discussion in computing?. In *Proceedings of the seventh international workshop on Computing education research.* ACM, 45–52. http://dl.acm.org/citation.cfm?id=2016923

[61] Sharon M Ravitch and Nicole Mittenfelner Carl. 2019. *Qualitative research: Bridging the conceptual, theoretical, and methodological.* SAGE Publications, Incorporated.

[62] Paul J Riccomini, Gregory W Smith, Elizabeth M Hughes, and Karen M Fries. 2015. The language of mathematics: The importance of teaching and learning mathematical vocabulary. *Reading & Writing Quarterly* 31, 3 (2015), 235–252.

[63] Jean Jinsun Ryoo. 2013. *Pedagogy Matters: Engaging Diverse Students as Community Researchers in Three Computer Science Classrooms.* Ph.D. Dissertation. UCLA.

[64] Carsten Schulte. 2008. Block Model: An Educational Model of Program Comprehension As a Tool for a Scholarly Approach to Teaching. In *Proceedings of the Fourth International Workshop on Computing Education Research* (Sydney, Australia) *(ICER '08)*. ACM, New York, NY, USA, 149–160.

[65] Thomas A Schwandt et al. 1994. Constructivist, interpretivist approaches to human inquiry. *Handbook of qualitative research* 1 (1994), 118–137.

[66] Sue Sentance, Jane Waite, and Maria Kallia. 2019. Teachers' Experiences of using PRIMM to Teach Programming in School. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education.* 476–482.

[67] Sue Sentance, Jane Waite, and Maria Kallia. 2019. Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education* 29, 2-3 (2019), 136–176.

[68] Karim Shabani. 2016. Applications of Vygotsky's sociocultural approach for teachers' professional development. *Cogent Education* 3, 1 (Nov. 2016). https://doi.org/10.1080/2331186X.2016.1252177

[69] Andrew Shenton. 2004. Strategies for Ensuring Trustworthiness in Qualitative Research Projects. *Education for Information* 22 (2004), 63–75. https://doi.org/10.3233/EFI-2004-22201

[70] John McHardy Sinclair and Malcolm Coulthard. 1975. *Towards an analysis of discourse: The English used by teachers and pupils.* Oxford Univ Pr.

[71] Art Sloan and Brian Bowe. 2014. Phenomenology and hermeneutic phenomenology: The philosophy, the methodologies, and using hermeneutic phenomenology to investigate lecturers' experiences of curriculum design. *Quality & Quantity* 48, 3 (2014), 1291–1303.

[72] Josh Tenenberg and Maria Knobelsdorf. 2014. Out of our minds: a review of sociocultural cognition theory. *Computer Science Education* 24, 1 (Jan. 2014), 1–24. https://doi.org/10.1080/08993408.2013.869396

[73] Jennifer Tsan, Collin F. Lynch, and Kristy Elizabeth Boyer. 2018. "Alright, what do we need?": A study of young coders' collaborative dialogue. *International Journal of Child-Computer Interaction* 17 (Sept. 2018), 61–71. https://doi.org/10.1016/j.ijcci.2018.03.001 00019.

[74] Jan Vahrenhold, Quintin Cutts, and Katrina Falkner. 2019. Schools (K–12). In *The Cambridge Handbook of Computing Education Research*, Sally A. Fincher and Anthony V.Editors Robins (Eds.). Cambridge University Press, 547–583. https://doi.org/10.1017/9781108654555.019

[75] Anne Venables, Grace Tan, and Raymond Lister. 2009. A Closer Look at Tracing, Explaining and Code Writing Skills in the Novice Programmer. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop* (Berkeley, CA, USA) *(ICER '09)*. ACM, New York, NY, USA, 117–128.

[76] Lev S Vygotsky. 1962. Thought and word. In *Studies in communication. Thought and Language*, Lev S Vygotsky, E. Hanfmann, and G. Vakar (Eds.). MIT Press, 119–153.

[77] Lev S Vygotsky. 1978. *Mind in society.* Cambridge, MA: Harvard University Press.

[78] Lee S. Vygotsky. 1981. The Instrumental Method in Psychology. In *The concept of activity in Soviet psychology*, J. V Wertsch (Ed.). Armonk, NY, Sharpe.

[79] Lev S Vygotsky. 1991. Genesis of the higher mental functions. *Learning to think* (1991), 32–41.

[80] Aida Walqui. 2006. Scaffolding instruction for English language learners: A conceptual framework. *International Journal of Bilingual Education and Bilingualism* 9, 2 (2006), 159–180.

[81] James V Wertsch. 1996. Mediation. In *Introduction to Vygotsky*, Harry Daniels (Ed.). Routledge, 1–34.

[82] James V Wertsch and Peeter Tulviste. 1992. LS Vygotsky and contemporary developmental psychology. *Developmental psychology* 28, 4 (1992), 548. 00552.

[83] R Paul Wiegand, Anthony Bucci, Amruth N Kumar, Jennifer L Albert, and Alessio Gaspar. 2016. A data-driven analysis of informatively hard concepts in introductory programming. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education.* ACM, 370–375.

[84] Ian Wilkinson and Kathryn Nelson. 2019. Role of Discussion in Reading Comprehension. In *Visible Learning Guide to Student Achievement: Schools Edition*, John Hattie and Eric M Anderman (Eds.). Routledge, 231–237.

[85] Jeffrey W Wimer, Carolyn S Ridenour, Kelli Thomas, and A William Place. 2001. Higher order teacher questioning of boys and girls in elementary mathematics classrooms. *The Journal of Educational Research* 95, 2 (2001), 84–92.

[86] Pelin Yüksel and Soner Yıldırım. 2015. Theoretical frameworks, methods, and procedures for conducting phenomenological studies in educational settings. *Turkish online journal of qualitative inquiry* 6, 1 (2015), 1–20.

[87] Daniel Zingaro. 2014. Peer Instruction Contributes to Self-efficacy in CS1. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 373–378. https://doi.org/10.1145/2538862.2538878