Learning Practice and Theory in Programming Education

Students' Lived Experience

Anders Berglund
Uppsala Computing Education Research Group, UpCERG
Uppsala University
Uppsala, Sweden
Anders.Berglund@it.uu.se

Anna Eckerdal
Uppsala Computing Education Research Group, UpCERG
Uppsala University
Uppsala, Sweden
Anna.Eckerdal@it.uu.se

***Abstract*— In learning to program, there is a complex interplay between the learning of practice (what to write, how to read compiler messages, etc.) and the learning of theory (what programming constructs are good for, how they "work", how the programme executes etc.). Our on-going project focuses on this interplay - normally not directly visible for the learners themselves - and offers insights about the complexity of the learning process. From a micro-level analysis of video films of students' collaboration in lab sessions, we follow how the students attention moves from one aspect to another (theoretical or practical), and then further, until, in good cases, we can document that a meaningful learning has taken place. Theoretically the project takes its point of departure in a framework, inspired by the roots of phenomenography and variation theory. Theory and practice are here interpreted from a pragmatic perspective, close to the students' (and, as we believe, most programmers) intuitive understanding and use of the terms. The ultimate aim of the project is to support teachers and teaching institutions to teach programming in such ways that students better learn how to program. That is, the project takes its point of departure in the disciplinary learning - the learning of programming - and aims to propose possible improvements in our teaching, by building on the insights gained by the micro-analyses of the students' learning. In this paper we illustrate, from an example, how the research process leads us from micro-level observations on how the students attentions move from different aspects (both theoretical and practical) of the half-ready computer program they aim to finalize, over our analysis of the lab sessions, to some insights in the complex interplay between theory and practice in programming students' learning process.**

Keywords— Learning to program, theory and practice, phenomenography, variation theory, micro-analysis

I. INTRODUCTION

As experienced programming teachers, we are aware that we need to encourage the students to learn both the theoretical aspects of programming (for example the meaning of an if-clause, the idea methods) more or less at the same time as they learn the practical aspects (the handicraft, such as where to put semicolon, how to act on error messages), in order for a good learning outcome. Our experience says that one of these aspects, taught without the other, will give an inferior learning outcome, possibly even to fail.

These insights, seen in the light of relevant research results in learning in STEM (Science, Technology, Engineering and Mathematics) and the phenomenographic theoretical framework of learning, stimulated us to formulate a research project, which we call *Theory and practice in lab work - a complex interplay* [2], on the complex interplay between theory and practice in programming education.

The ultimate aim of the project is to support teachers and teaching institutions to teach programming in such ways that students better learn how to program. That is, the project takes its point of departure in the disciplinary learning - the learning of programming - and aims to propose possible improvements in our teaching, by building on the insights gained by the micro-analyses of the students' learning. We also believe that the results will be useful in other disciplines with laboratory work.

II. RESEARCH QUESTION

With this as a background, we address the following research question in the current paper:

- What is the role of practice when students learn to program in terms of theoretical knowledge and practical skills in the lab?

We illustrate, from an example how our research process, starting with micro-level observations on how the students' attentions, moves between different aspects (both theoretical and practical) of the half-ready computer program they aim to finalize during their lab sessions. We then analyse the data from the lab session to continue with a preliminary discussion on some results concerning how theory and practice interact when students learn to program in the lab.

III. RELATED WORK

The present work focuses on the relationship between learning of theory and learning of practice in the lab. We thus draw on previous research on the distinction between theory and practice, and on how students learn in the lab.

There exists no universal definition of what is meant by 'theory' and 'practice'. In educational research in the Western

culture there seems to be an accepted agreement that theory and practice are opposite parts of a dualistic opposition [1], but different terminologies are used with somewhat different meanings in different research traditions. From higher education research [3] discuss WTP, "ways of thinking and practicing". This concept highlights the fact that competence in a subject area not only involves the ability to master certain subject-specific ways of thinking, but also the ability of practicing. There exists a considerable body of research in mathematics education research where knowledge is largely divided into two types, often referred to as 'conceptual' and 'procedural' and similar to the distinction between theory and practicing made here. McCormick [4] writes that these concepts relate to "a familiar debate in education, namely that of the contrast of content and process (p149) ... In mathematics education the argument has been about 'skills versus understanding'." From technology education [4] makes a similar distinction, discussion 'conceptual knowledge' and 'procedures', while from physics education von Aufschnaiter and von Aufschnaiter [5] discuss how students learn 'theory' and 'practice' in the lab.

Learning by practicing programming, for example in the lab, is an important and not questioned part of programming educations Computer Science Curriculum [6]. Höök [7] studied some factors that correlated to final exam results in an introductory programing course. The factors were e.g. how many hours the students had coded themselves, and how many hours they had watched a peer student code while they solved a problem together. Höök found that the students with the highest grades on the exam had spent considerable more time coding themselves compared to students with lower grades. The exam included questions on writing, reading code, and some questions on theory. Even though previous research as well as established experiences point to how important it is to practice, little is known on what happens during practical lab work in terms of the learning of theory and practice, and why it is so important for students to practice.

Although students are normally offered exercises and opportunities to practice in the lab under supervision in programming courses the failure and dropout rates are still high ([8], [9], [10], [11], [12], and [13]). The main focus to improve learning of programming has been on learning technologies and techniques like developing and evaluating software tools ([14], [15], and [16]). Still the problems remain.

Some research from computing education has focused on how students learn theory in terms of concepts. Here the main focus has been on misconception ([17], [18]). Examples of research on students' understanding of concepts are [19], [20], and [21].

The problematic relationship between learning theory and learning practice in the lab is pointed to in areas like natural science and technology ([22] [23], [5], and [4]). Only little is written on this relationship in programming education. Holmboe [24] emphasises that good understanding in programming requires both practical skills and conceptual understanding, and a connection between the two. The complex relationship between conceptual learning and practice has been recognized by for example du Bolay [25] who discusses domains that programming students must learn to master. These include the syntax and semantics of a programming language and different programming skills. du Boulay writes:

> None of these issues are entirely separable from each other, and much of the 'shock' [...] of the first few encounters between the learner and the system are compounded by the student's attempt to deal with all these different kinds of difficulty at once. (p. 284).

IV. THEORETICAL UNDERPINNINGS

The project leans theoretically on the phenomenographic research framework. Phenomenography often is seen, and used, as a methodology, aiming to unfold and describe the qualitatively different ways in which students understand, or relate to, something. In this project, however, we build on its theoretical underpinnings that offers a framework for distinguishing a certain aspect of something from another aspect of the same phenomenon using a second-order perspective, as well as the ideas underlying variation theory, stressing the necessity of variation in what the students see, or experience, for a meaningful learning to take place.

The concept of Object of Learning also has its roots in phenomenography. Marton & Booth [26] describe it in the following way: "Object of Learning [can be] described on the collective level, as a complex of the different ways in which the phenomenon can possibly be experienced from the point of view of individual learning." (p. 163, slightly edited) That is, the object of learning, in this interpretation, defines what can be learned about something in a certain setting.

The object of learning has three facets, relevant for this project and that varies over what it describes and by whom it is seen. The intended object of learning is the aim of the teaching, as seen by the teacher, the enacted object of learning describes what is possible to learn (not to be confused with the learning goals, a term from an administrative perspective and formal course plans), from the perspective of a researcher, while the lived object of learning is the learning outcome at a certain point of time, as seen by the learners themselves. ([27], and [28])This paper takes the students' perspective and thus has its focus on the lived object of learning.

In the analysis we refer to gaps in students' understanding in the sense discussed by Wickman and Östman [29] and Lidar, Lundqvist and Östman [30]: "Learning is a process where gaps are filled by construing new differences and similarities in relation to what is immediately intelligible" [29, p. 603]. Lidar et al. [30] write "The concept of gap is used to operationalize situations where people in action show that they are trying to make the activity they are engaged in to proceed." (p. 152)

Theory and practice

There is no universal understanding of the concepts *theory* and *practice*, as we have discussed above. In the present analysis we rather discuss theory-oriented actions, and practice-oriented, following [2]:

- When we are able to use language […] to adequately express an intended meaning that corresponds to our

present understanding of a learning object, then we express knowledge in a theory-oriented way.

- When we are able to act adequately towards an intended end, based on our present understanding of a learning object, then we express knowledge in a practice-oriented way.

V. The Empirical Setting and Data Collection

Data for the T-PIPE was collected from programming teachers and students in upper secondary schools (high schools) as well as at undergraduate university level. The data that are relevant for this paper stems from students, taking their first programming course, doing a lab in their first programming course at three universities in Sweden. The intended learning outcomes were similar at the three universities: methods that return value in Java, or functions in Python, but the details how the lab was constructed and what had been taught previously to the lab varied slightly between the universities. This influenced which details the students focused on and were stuck on during the lab.

We video filmed 29 students distributed over 13 groups, with each group consisting of two to three students collaborating at one computer. Each group was filmed from behind in order for us to see what gestures or movements the students did (for example pointing to the screen) and to be able to distinguish between the two or three students. We also captured the screens and the sounds to be able to follow in detail what the students did and their conversation. After the lab individual stimulated recall interviews with the students were performed where we discussed the lab. We showed them short passages from the films, in which we judged that the student had gained new insights and asked him or her to describe what happened. We thus have two films, one from a camera and one from the screen, from each student group as well as the audio recordings of the individual interviews, plus the audio recorded individual interviews. The interviews and the screen films were transcribed verbatim.

VI. The Analysis

During the analysis we looked for sequences in the data where it seemed as if both, or one of, the students gained some new insights. Inspired by [31] we refer to these sequences as threads of learning. The topics of discussion in the threads of learning varied, but were always related to the lab and its purpose. It was often a technical detail (for example, the role of curly brackets, which was not explicitly mentioned as a learning goal, but that still is a requirement for managing the larger task, to the purpose of methods that returns value in Java or functions in Python.

The possible threads of learning were, in most cases, identified by the second author, and were then discussed by the two authors, until a consensus was reached on possible interpretations of the thread of learning or until the possible thread was jointly judged as unproductive from the point of view of the research questions asked in the project.

The analysis was explorative in its nature. We, as researchers, at certain occasions took the perspectives of the students, that is, a view from within the data where we were aiming to come close to see what the students saw. At other occasions, but still on the same episodes, we took an outside perspective, being analytic and analysing what, of the topic at hand that could be learnt. Methodologically we followed the four-step model described in [32], which has been developed as a part of this research project.

In our analysis in the present article we studied one thread in relation the other possible threads, looking for differences and similarities. Further, we followed the "flow" or the thread of the discussion to see how it moved between topics that had a focus on practice and on those that focused on theory.

VII. An Example of the Interplay Between Theory and Practice

In this article we use an example of a thread of learning from our data to illustrate the close relationship between students' learning of theory, referred to in this text as theory-oriented action and their learning of practice, here referred to as practice-oriented action. Our conclusions, presented in context of the example, are based on our analysis of several similar examples, and are thus not solely based on this example.

In our example we see how the students move between focusing on theory and focusing on practice in what they do. When our example starts, the three students, Adam, Bengt and Cecilia[1] , are supposed to write an if-statement in a method. When reading the supporting on-line document required for the task, they realize that they do know neither the syntax nor the semantics of curly brackets in Java. Their lived object of learning has thus narrowed from being on how to write methods in Java, to the if-statement, to now on how to use and write curly brackets in Java. Despite this not being stated explicitly as the intended object of learning, it is still a relevant learning task, since writing methods in Java always involves using curly brackets.

Our example is structured on the following way: Firstly, in section A, we set the scene by presenting the situation, showing the code that the students have written up until now and the examples from the supporting document that they have found. Secondly, in section B, we listen to the discussion between the students, with our annotations on the interplay between theory and practice intertwined between the statements of the students. Thirdly, in section C, we summarize how we see the students moving between theory and practice.

A. *The scene for the example*

The students' code, written in the editor, looks like this when our example begins. Their code is not correct. The lines are numbered for later reference.

```
1. public void distClosestWall() {
2.     this.getModelDisplay().getHeigth();
```

[1] The names used in this paper are not the real names of the students, and do not reflect their gender

```
3.      this.getModelDisplay().getWidth();
4.      if (getHeight()<getWidth());
5.      System.out.println(int getHeiget());
6.           else
7.      System.out.println(int getWidth());
8.  }
```

At this point the students are not sure if there should be a semicolon after `else` on line 6. They open the supporting document with the explanation and find examples of the if-statement. The examples the students study are showed below:

Example 1 from the supporting document

```
if (x > 0) {
    System.out.println(“Positive.”);
}
```

Example 2 from the supporting document

```
if (x > 0) {
    System.out.println(“Positive.”);
}
else {
    System.out.println(“Not positive.”);
}
```

B. The students discussions with our annotations

The students now become aware of differences between their own code and the examples in the supporting document. They start to discuss curly brackets. Text in italics is our comments from our observations from the film taken from behind (see section V).

1. *The students are writing an if-statement and discuss whether they need semicolon after else. They open the document on if-statements and start to read it.*

The sequence starts with a practice-oriented action (1).The work continues:

2. Adam: They have these *points to curly brackets* after those. Should you then … *silence*. Where is it then? *Cecilia points to where { and } are at the keyboard*
3. Cecilia: How often are you supposed to use them?
4. Adam: Should we use it *Referring to the curly bracket* after if too? *Adam refers to line 4.*
5. *Adam types { directly after the keyword if*
6. Bengt: Hm. Let’s first try to figure that out.
7. *Adam presses Enter after the first print-statement on line 5 and types } on the new line*
8. Cecilia: I would have put it before the if so if kind of comes inside.

The practice-oriented action (1) triggers a theory-oriented discussion (2-4, 6, 8). During this, Adam’s practice-oriented actions of typing brackets (5, 7) lead to further theory-oriented discussions (6, 8). Their work continues:

9. Bengt: They don’t do it like that there. *Referring to the examples in the document*

Since the discussion does not lead the students to an answer to how curly brackets are used and why, Bengt continues by reading further in the document (9). This helps Adam to advance:

10. Adam: Ops, it shouldn’t be there, it should be here.
11. *Adam moves { to the end of line 4, after the if-statement and the semicolon.*
12. Cecilia: What, why did we put it there?
13. *Adam types { after else on line 6*
14. Bengt: Because they did it like that. *Bengt refers to the examples in the supporting document*
15. Cecilia: But why do you do like that? I image that it kind of does, what do those brackets do? I image that it should be like, parenthesis you kind of do first.
16. *Adam types { after the print-statement on line 7.*
17. Bengt: No it’s not parenthesis, it has nothing to do with that. *Bengt sounds very sure*.
18. Cecilia: But why do you put, what is it really?
19. Bengt: I think it’s the end of the statement. This is the statement.
20. Adam: Yes, it’s the part we are working on. This is the if-statement. *Adam points to the screen to the statement that is connected to if.* This is kind of the method that is included in if. This is what makes if work.

Theory-oriented activity	*Practice-oriented activity*	*Comment*
	Write a method. Open the document on if statements.	The method requires an if-statement.
(1) Read the document on the if-statement.		Discover [1] in the examples given.
	(3-4) How often…? Should we use it after if …?	Practice-oriented discussion on where curly brackets should appear, syntax.
	(5, 7) Adam types curly brackets	The action triggers a theory-oriented question:
(8) I would have put it before the if…		This comment triggers a practice-oriented action:
	(9-11) Ops, it shouldn't be there it should be here. [Moves the curly bracket {]	Pattern recognition in the document triggers the action, which starts a new discussion:
(12-21) What, why did we put it there? … what do those brackets do? … I image that it should be like, parenthesis		Now the students discuss the semantics of curly brackets and compare to previous knowledge.

Table 1. The table summarizes key points of the analysis regarding the interplay between theory-oriented and practice-oriented activities that occurred during the short sequence in the lab described above.

21. Cecilia: Aha, I understand. Okay, I get it.

This helps Adam to correct the position of the curly brackets (11) and place the other brackets (13, 16). A continued theory-oriented discussion of the semantics (12-21) is initiated by the practice-oriented action in (11). The theory-oriented discussion starts with a question from Cecilia (12), which makes Bengt points to the syntax "Because they did it like that" referring to the examples. Cecilia seems not satisfied and makes a comparison to previous knowledge on the semantics of parenthesis in language in general. Finally Bengt and Adam give suggestions on the semantics of curly brackets in if-statements. After this Cecilia seems satisfied: "Ok, I get it." This discussion seems to develop their understanding of the meaning and use of curly brackets even though it seems as if they believe that the brackets are needed for the if-statement to work at all (20-21).

After this sequence the code in the editor looks like this:

```
public void distClosestWall() {
    this.getModelDisplay().getHeigth();
    this.getModelDisplay().getWidth();
    if (getHeight()<getWidth()); {
    System.out.println(int getHeiget ());
    }
    else {
    System.out.println(int getWidth());
    }
}
```

The analysis is summarised and illustrated in Table 1. The table has three columns. The left column summarizes the theory-oriented activities we have identified in our analysis, and the middle column similarly contains a summary of the practice-oriented activities identified. The right column contains comments to explain and highlight what happens during this thread of learning, with a focus on how the theory- and practice-oriented activities interact and contribute to the full learning process.

C. Summary of the interplay between theory and practice in the example

To summarise, this thread of learning in the lab session gives the student a possibility to work with, and learn about, curly brackets in relation to if-statements when developing the method `distClosestWall()`.

The questions raised relate to the syntax and semantics of curly brackets: how are they used, what is the purpose of them, and what do they correspond to in relation to the students' previous knowledge? The students come up with several different suggestions. Eventually they put the curly brackets in the correct positions, after comparing with examples in a document, and Adam gives an explanation on why they are used. The students seem to have got a first glimpse of how curly brackets embrace statements that belong to the if-statement. They do not seem to understand neither the syntax completely (they still have a semicolon in the end of line 4), nor the full meaning of curly brackets, nor do they have a relevant language to express their current understanding (Adam calls the body of the if-statement "the method that is included in if").

The variation in the syntax lies in that the code in the examples in the supporting documents differ from their own code in that they do not use curly brackets. This insight triggers a movement in their attention. Further there is a variation between what Adam reads in the document and where he first puts the left bracket { directly after if. Cecilia finally tries to compare curly brackets to ordinary parenthesis to better understand their semantics. This variation in the syntax comes to the fore by practice-oriented actions as well as theory oriented reflections: reading the document and typing a curly

bracket in a way that seems correct, and reflecting on how this relates to ordinary parenthesis.

The discussion of syntax precedes that of the semantics, with suggestions from students' pre-knowledge in-between. Further, it is only one of the students who strives to develop her conception through understanding the meaning of curly brackets, the other two seem to, which they also say in the subsequent interviews, strive for pragmatic solutions, to write a syntactically correct if-statement.

VIII. CONCLUSIONS

The focus in the TPIPE-project is to get a better understanding of the complex relationship between students' learning of theory and their learning of practice. In this section we discuss some patterns on how theory and practice interact in students' learning that our example illustrates and that we can see in our larger data set.

The example from the on-going research discussed in this paper illustrates a common pattern in our data: It is not until the students do something in practice; here they start to write an if-else-statement, that they notice a variation between their own code and the code in the examples. In the practice the students open up a variation so that they become aware of a gap in their understanding; they do not know the syntax and semantics of curly brackets in Java. When the students notice the gap they try to fill it. A complex movement between practice-oriented and theory-oriented actions follows. The pattern we see is:

- In the practice students notice variation, often created by the students themselves, which makes them becoming aware of a gap in their understanding. The practice thus seems to strengthen and direct students' attention, through variation, to such gaps and thus opens for learning.

Another observation from this example is how the students go about trying to fill the gap. First they focus on the syntax rules. In the example, Bengt answers Cecilia's question why they put the curly bracket at a certain position: "Because they did it like that", referring to the document they are studying. In the next phase Cecilia, who doesn't seem to be satisfied with this answer attempts to compare and connect the use of curly brackets with ordinary parenthesis. Cecilia's suggestion is firmly dismissed by Bengt. This leads however to a further discussion on the semantics of curly brackets where Adam and Bengt contribute with some aspects. Situations like this, with attempts to connect to previous knowledge are discussed by [30]. The authors write: "In order to fill the gap, relations have to be created between what the students already know and what is new in the situation." (p. 152). The pattern that we see here and elsewhere in our data indicates that the attempt to fill a gap in programming might often follow a certain pattern:

- Student often first focus on syntax and how to do in practice. If they do not succeed, a theoretical discussion may start where the focus shifts to previous knowledge and whether this can contribute to filling the gap. Finally, if this attempt did not succeed, a continued theoretical discussion on what is new in the situation, or practical attempt, may follow.

An initial practice-oriented action, which can be characterized as for example 'we do as we did in the previous exercise'-thinking, or 'this is what the lab instruction says we should do'-thinking, or 'I just happened randomly do this', can trigger a need for understanding why, or why not, it should be done in this particular way. A theory-oriented discussion, which in our data is often focused on semantics, may follow to resolve the problem. When the students seem content with their explanation, or if they cannot come further, they continue with a practice-oriented action. Again, the practice might make the students aware of a gap in their understanding which leads to a theory-oriented discussion. Thus, we can follow a wave, or a movement forward in the learning process, which alters between practice-oriented actions and theory-oriented reflections, but where the practice-oriented actions frequently, but not always, seem to be the initial triggers of the movement. This may be because the students' own practice creates variation, which helps them discover gaps in their knowledge/understanding.

Already before this project started we were aware of that both theory and practice were needed for a good learning of programming. But how closely connected these two facets are, to what extent they support each other and the whole, and how many ways in which they can interact, has only became visible to us as during this research project.

ACKNOWLEDGMENT

We want to thank the students who participated in this study, and our colleagues Lennart Rolandsson, Inga-Britt Skogh, and Michael Thuné. This work is supported by a grant from the Swedish Research Council (Vetenskapsrådet), for which we are grateful.

REFERENCES

[1] B. Molander, *Kunskap i handling*: Daidalos, 1996.

[2] A. Eckerdal, "Theory and practice in lab work - a complex interplay," Grant application to Vetenskapsrådet (the Swedish Research Council) 2011.

[3] V. McCune and D. Hounsell, "The development of students' ways of thinking and practising in three final-year biology courses," *Higher Education*, vol. 49, pp. 255 - 289, 2005.

[4] R. McCormick, "Conceptual and Procedural Knowledge," *International Journal ofTechnology and Design Education*, vol. 7, pp. 141 - 159, 1997.

[5] C. von Aufschnaiter and S. von Aufschnaiter, "University students' activities, thinking and learning during laboratory work.," *European Journal of Physics*, vol. 28, pp. 51 - 60, 2007.

[6] "Computer Science Curriculum," Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM), IEEE Computer Society2013.

[7] L. J. Höök, "On the bimodality in an introductory programming course: an analysis of student performance factors. ," Department of Information Technology, Uppsala University, Uppsala2014.

[8] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, *et al.*, "A multi-national, multi-institutional study of assessment of programming skills of first-year CS students," *SIGCSE Bull.*, vol. 33, pp. 125--180, 2001.

[9] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer Science Education*, vol. 13, pp. 137 - 172, 2003.

[10] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, *et al.*, "A multi-national study of reading and tracing skills in novice programmers," pp. 119--150, 2004.

[11] A. Eckerdal, R. McCartney, J. E. Moström, M. Ratcliffe, K. Sanders, and C. Zander, "Putting Threshold Concepts into Context in Computer Science Education," *SIGCSE Bulletin inroads*, vol. 38, pp. 103-107, 2006.

[12] P. Kinnunen and L. Malmi, "Why students drop out CS1 course?," in *Second international workshop on Computing education research (ICER '06)*,, 2006, pp. 97 - 108.

[13] P. Kinnunen, "Challenges of teaching and studying programming at a university of technology: Viewpoints of students, teachers and the university. ," Helsinki University of Technology, Helsinki, Finland., Unpublished PhD thesis 2009.

[14] P. Gross and K. Powers, "Evaluating assessments of novice programming environments," pp. 99--110, 2005.

[15] K. Powers, S. Cooper, K. J. Goldman, M. Carlisle, M. McNally, and V. Proulx, "Tools for Teaching Introductory Programming: What Works?," *ACM SIGCSE Bulletin*, vol. 38, pp. 560 - 561, 2006.

[16] D. Valentine, "CS educational research: a meta-analysis of SIGCSE technical symposium proceedings," in *the 35th SIGCSE technical symposium on Computer science education*, Norfolk, Virginia, USA, 2004, pp. 255-259.

[17] N. Ragonis and M. Ben-Ari, "A long-term investigation of the comprehension of OOP concepts by novices," *Computer Science Education*, vol. 15, pp. 203-221, Sept. 2005.

[18] K. Sanders and L. Thomas, "Checklists for grading object-oriented CS1 programs: concepts and misconceptions.," in *Innovation and technology in computer science education (ITiCSE '07). ,* 2007.

[19] A. Berglund, *Learning computer systems in a distributed project course: The what, why, how and where* vol. 62. Uppsala, Sweden: Acta Universitatis Upsaliensis, 2005.

[20] C. Bruce, L. Buckingham, J. Hynd, C. McMahon, M. Roggenkamp, and I. Stoodly, "Ways of experiencing the act of learning to program: A phenomenographic study of introductory programming students at university," *Journal of Information Technology Education*, vol. 3, pp. 143--160, 2004.

[21] A. Eckerdal and M. Thuné, "Novice Java programmers' conceptions of "object" and "class", and variation theory," *SIGCSE Bulletin inroads*, vol. 37, pp. 89–93, 2005.

[22] A. Hofstein and V. N. Lunetta, "The Laboratory in Science Education: Foundations for the Twenty-First Century. ," *Science Education*, vol. 88, pp. 28 - 54, 203.

[23] M.-G. Séré, "Towards Renewed Research Questions from the Outcomes of the European Project Labwork in Science Education," *Science Education*, vol. 86, pp. 624 - 644, 2002.

[24] C. A. Holmboe, "Cognitive Framework for Knowledge in Informatics: The Case of Object-Orientation," in *Proceedings of the 4th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education*, 1999.

[25] J. du Boulay and H. Benedicte, "Some difficulties of learning to program," 1989.

[26] F. Marton and S. Booth, *Learning and awareness*. Mahwah, New Jersey, USA: Lawrence Erlbaum Associates, 1997.

[27] F. Marton and A. B. M. Tsui, *Classroom Discourse and the Space of Learning*. Mahwah, New Jersey: Lawrence Erlbaum Ass., 2004.

[28] F. Marton, *Necessary Conditions of Learning*: Routledge, 2015.

[29] P. O. Wickman and L. Östman, "Learning as discourse change: A sociocultural mechanism.," *Science Education*, vol. 86, pp. 601 - 623, 2002.

[30] M. Lidar, E. Lundqvist, and L. Ostman, "Teaching and learning in the science classroom - The interplay between teachers' epistemological moves and students' practical epistemology," *Science Education*, vol. 90, pp. 148-163, Jan 2006.

[31] Å. Ingerman, C. Linder, and D. Marshall, "The learners' experience of variation: following students' threads of learning physics in computer simulation sessions. ," *Instructional Science*, vol. 37, pp. 273 - 292, 2007.

[32] A. Eckerdal and M. Thuné, "Analysing the enacted object of learning in lab assignments in programming education," presented at the Proceedings of the 2013 Learning and Teaching in Computing and Engineering conference, Macau, 2013.