International Journal of Teaching and Learning in Higher Education 2017, Volume 29, Number 2, 245-254
http://www.isetl.org/ijtlhe/ ISSN 1812-9129

An In-Depth Analysis of Teaching Themes and the Quality of Teaching in Higher Education: Evidence From the Programming Education Environments

Belle Selene Xia
Aalto University

Education research in computer science has emphasized the research of web-based learning environments as a result of the latest technological advancement in higher education. Our research aim is to offer new insights on the different teaching strategies in programming education both from a theoretical and empirical point of view as a response to the theory-scarce nature of the subject. We have classified the teaching themes in computing education research based on the students' experience and reviewed the respective teaching methods introduced by the previous literature in the subject field. Our research results confirm that despite the benefits brought by technology to higher education and the high quality of the programming courses, there exist challenges associated with programming education environments that need to be addressed with further research. We bring up the concepts of student-centered pedagogy and personalized learning environments in response to the challenges faced by students in programming education. Specifically, we will analyze these challenges via teaching strategies and by considering the students' needs in a collaborative learning environment. Our research results are especially valuable to the understanding of the development of the programming education environment. We will open up new research opportunities in the quality management of distance learning.

As a result of technological advancement, research in web-based learning environments has become ever more important. Researchers in programming education have confirmed both the benefits and the potential of technology brought to pedagogy in higher education (Sadler-Smith, Down, & Lean, 2000). While both theoretical and empirical papers are published in the subject field, research in programming education environments is still theory-scarce (Fincher & Petre, 2004). In addition, Teague, Corney, Ahadi, and Lister (2012) showed that students start to struggle with the challenges associated with programming at the beginning of the course. This phenomenon leads to drop-outs and increasing difficulties with learning programming in the latter part of the course, and these challenges of programming education need to be dealt with further research.

Lewis (2010) states that the goal of programming education is to develop the students' programming competence and attitudes towards programming. The aim is to teach students to understand the logic behind programming. Specifically, Whalley and colleagues (2006) showed that students who learn programming successfully are able to produce correct codes and explain the purpose behind these codes. Programming skills can be measured via the level of understanding in code-tracing tasks and the code-writing abilities, which are closely related to the code-explaining ability (Lopez, Whalley, Robbins, & Lister, 2008). Nevertheless, the relationship between the code-writing abilities and the code-explaining abilities is not directly related, as novice programmers are able to write codes based on familiar templates but may find it difficult to trace codes and correct the bugs (Simon, 2009). Previous research has established that the performance of novice students in systematically writing and explaining codes after introductory programming courses is reflected as minimal competence, and thus we will review some of the research on the challenges of programming education in the following section.

Research on the Challenges of Programming Education

Saeed, Yang, and Sinnappan (2009) found that the technological impact on higher education has brought challenges to teaching. The authors mention that one of the primary challenges associated with the use of technology in programming education is the lack of understanding of the learners' experience during such a process and their perceptions of the technology use in supporting their understanding. Below we have summarized some of the learning challenges in programming education based on the previous literature together with the teaching strategies in the respective situations. The research work done by Fincher and Petre (2004) has served as the foundation for the learning challenges specified in Table 1. It is seen that the typical learning challenges are related to knowledge sharing from the course instructor to the students having differing backgrounds. Students' motivation to learn and expectations may also pose challenges to the course organizers. The solutions to these challenges require us to identify and overcome these student misconceptions.

Student-Centered Pedagogy in Programming Education

Education research in computer science has emphasized the importance of studying the students' behavior as a gateway to improve the set of existing

teaching practices (Herrington et al., 2003). In other words, the teaching strategies to overcome these challenges in programming education are highly dependent on student-centered pedagogy. By studying the student experiences in programming courses, this offers important information on the design of programming education that supports student learning despite their individual differences in learning preferences. Koka and Hein (2003) observed that students' learning preferences and learning styles are affected by the teacher's feedback, learning challenges, and the intrinsic motivation to learn. Thomas, Ratcliffe, Woodbury, and Jarman (2002) found that successful learning outcomes can be explained via these different learning styles. The authors proposed that students' preference on learning and their expectations should be integrated in the design of programming courses via well-organized learning resources. This confirms the significance of our research, which is to evaluate programming education from the students' point of view.

Wolf (2002) showed that an interactive web-based adaptive learning environment, given its flexibility and dynamic nature, allows a personalized learning environment which accommodates different learning styles. A student-centered approach in teaching can be achieved through the application of andragogical assumptions. The assumptions of andragogy address the interests of the learners, cooperative learning, guided interaction, and the active role taken by the learner (Blondy, 2007). Chan (2003) suggests that the student performance can be enhanced, and some of the challenges of programming education, addressed by tailoring the programming environment according to the students' needs and individual working styles. Specifically, Bati, Gelderblom, and Biljon (2014) have found that engaging students for deeper learning, using support mechanism for improved class management, aligning assessment activities, and creating closer relationships and a sense of community among students are effective instructional strategies in programming teaching. According to the theory of constructivism, successful learning outcomes can be attained when learners' motivation is aligned with the teaching goals of the course and students can be motivated to engage in the learning process. As a matter of fact, the role of student has become more important in programming education design, as confirmed by researchers such as Herrington and colleagues (2003).

Further Investigation on the Themes of Teaching in Programming Education

Below we have classified the different themes of teaching in a technology-oriented learning environment based on the previous literature. It is seen that the different themes of teaching are reflected as a result of different teaching goals. Teaching methods that use technologies in higher education are especially significant in distance learning. Nevertheless, the integration of technologies to higher education does not always produce sound learning outcomes, and some of the advanced programming tools used in programming education seem to hinder learning when they are difficult to use. In addition to textual instructions, different visualization techniques are widely used in programming education, and well-designed multimedia messages are found to support learning. Teaching methods may also include cooperating learning and collaborative active techniques, such as pair programming, to enhance successful learning outcomes.

Distance learning. The use of computing technology and instructional design in learning has opened new opportunities to choose new innovative teaching methods. Especially distance learning has attained a higher importance in computer science education (Sadler-Smith et al., 2000).

Educational technology tools. Technological innovation has opened new opportunities for learning. Different educational technologies are utilized in teaching. Learning efficiency can be enhanced via appropriate educational technology tools (Clarke, Flaherty, & Mottner, 2001).

Technology innovations. Technology innovations are used to enhance students' learning experience. A measure of their success includes the extent of the skills developed by the students after the course. However, it is shown that technology innovations may not always enhance learning (Dacko, 2001).

Multimedia learning Multimedia learning uses words and pictures in learning. Compared to communication involving words, well-designed multimedia messages allow students to learn more deeply. Here, the design of multimedia explanation is the learning method used to achieve learning outcomes. Visualization techniques are extensively used in programming education (Mayer, 2003).

Peer feedback. The impact of using peers in the evaluation of student performance was found to be useful. Specifically, peer feedback was found to be meaningful and effective in higher education. A collaborative learning environment is also found to support programming education (Reese-Durham, 2005).

Case studies. Cooperating learning techniques can be combined with case studies in order to enhance problem-solving and decision making skills in learning. This type of learning technique is found to be more useful than lectures and non-cooperative learning (Baumberger-Henry, 2005).

Project-based learning. Teaching may be deductive or inductive. Deductive teaching method begins with a theory and then proceeds to the application of theories. Inductive teaching methods include inquiry

learning, problem-based learning, project-based learning, case-based teaching, discovery learning, and Just- In-Time-Teaching (Prince & Felder, 2006).

Cooperative learning. Compared to the traditional instructional learning approach, research results have shown that instruction based on cooperative learning yield significantly better achievement in terms of academic performance (Doymus, 2007).

Self-directed feedback. Mastering the learning technique of self-directed feedback, reinforcement, and remediation of knowledge is proven to have a positive effect on the transfer of knowledge, which is central to learning (Lee & Kahnweiler, 2008).

Collaborative active learning. When students first enroll to college and are not familiar with the course topic, collaborative active learning activities are found to be useful. Learning outcomes are measured as academic performance in terms of grades (Saitta, Gittings, & Geiger, 2011).

Statement of the Problem

In this study, we aimed to capture themes of teaching in programming education via student experiences and observations. Our research problem was formulated as follows: how do students perceive the different teaching themes of introductory programming courses as a response to the challenges of programming education? Specifically, our research questions were specified as follows:

1. Why do students find programming education difficult?
2. What are the themes of teaching that increase student motivation to learn programming?

Our research goal was to collect data from the students' behavior in programming education environments, analyze the data in order to identify various behavioral patterns in the student experience, and produce sound research results by evaluating how those teaching in programming education can use this information to produce sound learning outcomes. Our paper is organized as follows. First, we will justify the focus of our research based on the previous literature. Next, we will compare the results obtained from the literature review with the student data in order to deepen our understanding of student-centered pedagogy in programming education.

Method

Salinger, Plonka, and Prechelt (2008) emphasize that a qualitative research approach is especially useful in deriving meanings through conceptual description of a programming experience. Our aim in this paper is to capture new knowledge on programming education based on the students' experience. Therefore, qualitative research was found to be useful and appropriate with regard to our research question. As the method and validity of content analysis was heavily dependent on the researcher and the context in which the information was analyzed, we made judgements on the variations and approaches that were most suitable for our particular research problem in this study. We also discuss possible limitations and delimitations of the study.

Research Design

We used extensive student feedback collected via open-ended questions in this study as our primary empirical data. The amount and the quality of the data were chosen in accordance with the research question. Our interpretation of the data was made via inferences based on content analysis, which admittedly may result in some degree of bias. The interpretation of the outcome of data was done by two researchers. The analysis of programming education environments from the student perspective was done in stages. First, we started by using the existing learning resources on programming courses at Aalto University to collect student data. Then we developed an infrastructure that provided user modelling and personalization. Thereafter, we explored several ways to produce knowledge-based personalization of these student experiences derived from concept analysis and content indexing, which will be explained in the following sub-sections.

Sampling

We collected student feedback from the programming courses in Aalto University in Finland from the years 2009 to 2013 through open-ended questions and surveys to be analyzed by content analysis. The programming course is arranged every year in the spring and in the autumn. In 2013 the primary programming language for the course was Python. Specifically, this course had 4 hours of lecturers, 32 hours of self-learning, 77 hours of exercises, and 20 hours reserved for the exam and exam preparation. During the sample years, the lecture format stayed the same. The teaching goal of this course was to equip the course participants with understanding in the field and the skills of programming. The course materials included both printed materials and course book. This course also included an online forum where the course participants could communicate with the course organizers. The data were interpreted based on the outcome of the student feedback per course period. In terms of variables, the average student achievement level, initial expectation of the students, and student background, as well as the course instructor, may have

Table 1
Learning Challenges and Teaching Approaches in Programming Education

Programming Education Environments	
Learning Challenges	**Teaching Approaches**
The cognitive learning theory emphasizes the importance of individual differences in learning. These learning styles result in a student's unique learning preference. (Saeed et al., 2009)	Learning by doing and encourage knowledge integration such as helping students to organize their ideas are found to enhance coherent understanding. (Anzai & Simon, 1979)
Students adopt inappropriate attitudes and beliefs towards learning that interfere with the learning process and obtaining successful learning outcomes. (Lewis, 2010)	According to constructivism, learning involves the interpretation of information, and student attitude can be affected via a learner-centered approach to teaching. (Herrington, Oliver, & Reeves, 2003)
Programming misconceptions, such as linguistic misconceptions, arise from inappropriate transfer of knowledge. (Bayman & Mayer, 1983)	The instructor identifies learning misconceptions and their causes while devising ways to resolve them in a systematic manner. (Thota & Whitfield, 2010)
Previous programming experience and expectations of the course interfere with the motivation to learn and produce results. (Bonar & Soloway, 1989)	The instructor may provide interaction and social support for learning such as supervised lab activities and online collaborative discussion. (Blondy, 2007)
Computational models and syntax used in programming are difficult to understand especially for novice students. (Kahney, 1983)	The instructor may begin with simple and consistent computational models and use animations as an aid to learning algorithms. (Hundhausen, 2002)
The content and quality of the learning materials do not reflect the course goals nor do they assist students in grasping new knowledge. (Dacko, 2001)	Learning materials must support learning and the quality of the materials is reflected in learning outcomes and student performance in the exams. (Zuckerman, Arida, & Resnick, 2005)
The programming skills learned in school are context-dependent and cannot be automatically transferred and used elsewhere. (Csikszentmihalyi, 1991)	Programming skills enhance problem-solving skills and other skills which may be conducive to professional career. (Lopez et a., 2008)

changed per course base, which admittedly may have had an impact on the interpretation of results.

Procedures

Hopkins and King (2010) confirm the benefit of content analysis for social scientists as an effective method to analyze text data. One of the main benefits of content analysis is its allowance for empirical study of a social phenomenon through documentary text data. One of the primary goals in using content analysis in this study was to categorize text patterns and literature in an unbiased and reliable manner. Therefore, formal content analysis is used to make generalizations from the student feedback via classifications. According to the Heisenberg Principle, the very research process produces the potential for bias. When it comes to the reliability of the results, it is acknowledged that content analysis, just as other research methods, might result in some degree of bias. Nevertheless, as a research method, content analysis is a systematic and objective method of describing contextual information. The benefits of content analysis are its context-sensitive nature and flexibility in terms of research design.

In terms of the reliability and validity issues of the data analysis procedure and findings, the challenges of our research approach admittedly existed. First, reliable information was needed in the first place, for reliable analysis and student feedback may not always contain all the information needed to be studied. Using content analysis as the primary research method in the present study might have also resulted in some degree of researcher bias. We aimed to minimize the bias produced by the data and the methods via good scientific practice. Specifically, more than one researchers analyzed the collected data in order to reach

a final consensus on the outcome of the analysis. Moreover, we aimed to demonstrate the link between our research results and the data by describing the analysis process in detail. Furthermore, as the method and validity of content analysis is heavily dependent on the researcher and the context in which the information is analyzed, we made own judgments on what variations and approaches were most suitable for our particular research problem in this study.

Summary of Results

Areias and Mendes (2007) confirm that computer programming is difficult to learn and requires extensive work from students. According to student feedback, the level of difficulty is higher for students with no prior background in programming. Therefore, the designers of programming courses need to consider program design, the complex features of the programming language, and the lack of programming experiences among novice students. As learning programming involves formulating algorithms and transferring them to a programming language, understanding the syntax of the language and being able to execute and trace different program statements were especially challenging for students with no prior experience in programming. In addition to the challenges of learning programming that are classified in Table 1, research results show that students also had difficulties in installing and using the programming environment, understanding the role of programming constructs, learning the semantics of programming structures, and finding compilation errors based on the system feedback. These students did not seem to comprehend the strictness of the programming languages and the underlying notional machine. Therefore, selecting a strategy for an initial approach to teaching programming required us to understand the students' experience of the programming courses and what kind of learning resources students found helpful in learning programming.

In 2009, we collected extensive and detailed feedback from 461 students, in 2010 from 390 students, in 2011 from 363 students, in 2012 from 229 students, and in 2013 from 212 students. The student profiles included students from different departments. That is, the student profiles included both students with and without prior programming experiences. It is interesting to note that the student profiles not only included novice students, but also students who had studied more than five years in the same university.

In 2009, 61% of the respondents were satisfied with the demand of the course. In 2010 the percentage was 66%, in 2011 the percentage was 67%, in 2012 the percentage was 67%, and in 2013 the percentage was 72%. Thus, the student satisfaction towards the programming courses has steadily increased since 2009. The student satisfaction is reflected by the incremental improvements made in the course with regard to the quality of the lectures, course materials, supportive tools, and programming exercises. In 2009, the average grade given for the lectures was 2.82/4, the grade given for exercises was 2.98/4, the grade given for the materials was 3.16/4, the grade given for the exam was 2.73/4, and the grade given to the usefulness of the course was 2.78/4. Since then student satisfaction has increased with regard to how the course is organized. In 2013, the average grade given to the lectures was 2.87/4, the grade given to exercises was 3.36/4, the grade given to the materials was 3.33/4, the grade given to exam was 2.91/4, and the grade given to the usefulness of the course was 3.15/4.

We listed the different teaching strategies in Table 1 as a response to the typical learning challenges faced by the students. Patriarcheas and Xenos (2009) have found that some of these teaching strategies are significant in terms of the student participation and the creation of a personalized learning environment. The student experiences on the various themes of teaching described in Table 2 and Table 3 in terms of the course lectures, exercises, learning tools, and materials can be used to construct a personalized learning environment where student-centered pedagogy is emphasized to enhance the learning outcomes of programming education. Hopson, Simms, and Knezek (2001) has shown that the student-centered pedagogy in a technology-rich learning environment enhances high-order cognitive skills, which are required to learn programming. Moreover, the authors acknowledge that similarities and differences between online learning and the traditional classroom learning environment are most evident in terms of the course design, the level of interaction and the respective teaching effect on the students.

Table 2 summarizes the challenges associated with teaching of the introductory programming courses, including student motivation challenges and knowledge sharing failures, as we have shown in Table 1. Table 2 also lists the excerpts taken from student feedback in respective to the difficulties associated with the programming courses. When it comes to the course exercises, typical challenges were related to the time schedule and the varying level of difficulties of exercises. Other challenges associated with programming education included the mismatch between the student expectations and the teaching goals set by the course, as well as students having difficulties in synthesizing the topics to be learned. Table 3 summarizes the motivational themes associated with the programming courses together with the respective excerpts taken from feedback results. The themes associated with the well-designed programming courses included competent lecturers and effective course assignments. Active learning, hands-on activities, and materials having exemplary solutions helped students to

Table 2
Difficulties Associated with the Programming Courses

Format	Themes	Specifications	Student Feedback
Lectures	Focus	The lecture focused too much details on the basics.	*"The information conveyed in the lecture was not always useful to advanced students."*
	Clarity	The pace of the lecture was too fast leaving gaps unexplained.	*"The information conveyed in the lecture was not always clear and related to the core of the course."*
	Usefulness	Students skipped classes and learned directly from the book.	*"Many of the students have never gone to the course lectures."*
	Quality	The quality of the lecture was poor and demotivating.	*"The instructor was not very motivational in terms of the course atmosphere."*
Exercises	Time	The students were not always given enough time to complete all the course exercises.	*"There was not enough time to complete the exercises."*
	Instruction	The exercise instructions were not clarified in advance.	*"The exercise instructions were difficult to understand from time to time."*
	Difficulty	Some of the exercises were found to be too difficult, especially for novice students.	*"The exercise was too mathematically intensive for novices."*
	Expectation	The exercise did not respond to student expectations.	*"Some of the exercises were too long, and thus were not expected by some of the students."*
Tools	Usability	The programming tool was too difficult to use.	*"The programming tool was difficult to use and too detailed."*
	Purpose	The programming tool did not enable easy finding of bugs.	*"The programming tool did not enable easy finding of programming bugs."*
	Grading	The programming tool fined too harshly for small mistakes.	*"Some students felt that the programming tool had allocated the points in an unfair manner."*
	Feedback	The programming tool did not provide enough guidance.	*"The programming tool did not always give instructions on how to fix the bugs."*
Materials	Relatedness	The course material did not relate to the course exercises.	*"Some of the students did not use all of the materials provided by the course."*
	Content	The course material contained too much texts with no key points.	*"For advanced programmers, the course material contained too much information."*
	Demonstration	The course material lacked demonstrations and visual aids.	*"The course materials contained too much texts, which may in times hinder understanding."*
	Availability	The course material was not easily available.	*"The availability of all the course materials was not clear to all of the students."*

Table 3
Motivational Themes Associated With the Programming Courses

Format	Themes	Specifications	Student Feedback
Lectures	Lecturer	The lecturer was competent and knowledgeable.	*"The instructor is knowledgeable and presented the subject by considering the needs of the students."*
	Style	The lecture motivated students to participate and learn.	*"The examples and exercises were useful to go through with the instructor in a step-wise fashion."*
	Audience	The lecturer considered the background of the students.	*"For those novice students, the lecture was found to be well organized with a memorable beginning."*
	Interest	The lecture contained interesting materials not found in the book.	*"The instructor has gone through interesting materials not covered in the course."*
Exercieses	Level	The exercise level proceeded from easy to difficult.	*"The difficulty level proceeded logically from easy at the beginning and challenging at the end."*
	Hands on	The exercises enabled learning by doing, which was an optimal learning style for some of the students.	*"The exercise enabled learning by doing."*
	Goal	The exercise supported the teaching goal of the course.	*"The exercises had good instructions and supported the course goals."*
	Complexity	The exercise was complex enough to capture student interest.	*"The exercises were found to be interesting and varied with various levels of difficulty."*
Tools	Online	The programming tool supported distance learning and enabled students to earn course points.	*"The programming tool supported distance and online learning."*
	Consistency	The programming tool worked consistently without mistakes.	*"The programming tool worked consistently without mistakes."*
	Technology	The programming tool reflected advanced technology.	*"The programming tool reflected advanced technology and is one of the best course tools."*
	Importance	Students participated the course because of the programming tool.	*"The programming tool was one of the reasons why students participated in the course."*
Materials	Readability	The course material was clear to read with real-world problems.	*"The course materials were consistent and clear to read."*
	Concreteness	The information of the course materials was tailored to the needs of the students.	*"The course material showed how to code and debug programs."*
	Relevance	The course materials closely followed the lecture knowledge.	*"Specific information was relatively simple to find from the given material."*
	Example	The course material contained supportive examples.	*"In addition to the core information, exercise examples were found to be conducive to learning."*

practice their programming skills. In terms of the learning tools used in the programming courses, challenges and possibilities are both associated with the usability of these tools. Finally, it is important for the course materials to be concise and clear.

Discussion and Conclusion

Thompson (2008) defines learning programming as the process of understanding and applying programming knowledge to practice by solving computing problems in an innovative manner. Lister and colleagues (2006) found that successful programmers are able to produce innovative solutions to computing programs. Détienne and Soloway (1990) distinguished the techniques that experienced programmers use when trying to comprehend a program. When tracing a program and analyzing its execution to determine what operations occur and how its states change, experienced programmers may use either generic or specific values when tracing a program's execution. Thota and Whitfield (2010) introduced strategies to design introductory programming courses from constructivist and pedagogical points of view that address these challenges of programming education and student misconceptions via the available learning resources. In this study we found that the course instructor may address these student misconceptions by devising sound teaching strategies to overcome these challenges associated with programming education. In fact, some of the factors that affect programming education are known to affect education processes in general, but there are also specific

ones relevant to programming courses. These factors are, for example, prior attitude and programming experiences, materials and tools used to support programming, and the active involvement of students in the programming courses via learning by doing.

Universities have developed advanced tools to support programming education. Examples include TRAKLA2, JSav, UUhistle, jsParsons, and mobile parsons. In addition to the tools that are developed to support learning, virtual learning environments and learning resources have also been integrated into programming education. As examples, the A+ learning environment integrates a number of tools under the same user interface. Innovative learning resources have been introduced for course Programming 1, CSE-A1110. Likewise, Algoviz OpenDSA learning resources have been used for Data structures and algorithms courses (Helminen, Ihantola, Karavirta, & Malmi, 2012). By adopting these tools and learning environments we are able to collect data from their usage and get regular information on the user experience of these tools and environments. Specifically, we are able to get course and task evaluation results for the course participants, submission data, course quizzes, log data about how students interact with various assignments, and log data about how students read and interact with learning resources. These collected data can then be combined in a database in order to allow easy query. Using these data, it is possible to produce adaptive guidance to best resources, adaptive textbook, adaptive visualization, and adaptive feedback in order to improve the whole learning system through personalized guidance.

Our research goals is to collect data from the students' usage of programming education environments with regard to the quality of the course. We choose to analyze the data in order to identify various behavioral patterns among the students and provide feedback to the students regarding to the usage of these tools to support their studies. The quality of the course can be analyzed via the resources allocated to the course in terms of lectures, materials, supportive tools, and programming exercises. We investigate student behavior in both treatment and control group settings, as well as longitudinal settings (Brusilovsky et al., 2010). While studying data-driven personalization in IR and Recommendation Systems areas, we have seen that all kinds of recommender approaches and content analysis (LDA) research approaches are found to be useful. We have found that successful programming courses are well organized in terms of computing exercises and learning tools.

Maloney et al. (2004) specified that web-based learning tools support student-centered pedagogy. Fernandez and Sanchez (2003) found that the benefits of using these programming tools to support learning include the possibility to support students to study intuitively and visually. Specifically, Hundhausen (2002) found that the algorithm visualization technology is effective in programming education, offering learning exercises where students engage in visualization-related activities that are cognitively demanding. As a matter of fact, Zuckerman and colleagues (2005) stated that in teaching abstract problem domains, special learning elements and design materials with the purpose to foster learning are indispensable; examples include the use of multimedia messages and visualization techniques to support student learning.

Technological advancement has had a significant impact on higher education, especially from the teaching point of view. The challenges of programming education remain a popular topic of research; some of the challenges include poor progression and retention rates associated with introductory programming courses. We have found some of the possible explanations behind the poor progression and retention rates of introductory programming courses based on the student experiences in terms of the course lecturers, course exercises, the learning tools used in the course, and the course materials, as these themes have a vital impact on the student confidence, performance, and study habits in acquiring programming knowledge. In response to these challenges of programming education, Falkner and Falkner (2012) analyzed the student pedagogy from the social constructivist and community-based learning perspectives. The teaching methods used in constructivist learning, which are by nature collaborative, and the social aspect of constructivist learning enhance engaging and productive learning experiences as a result of group learning. We have confirmed in this study that collaboration in a programming environment via, for example, pair programming is vital and enhances learning efficiency. In terms of future research, it would be interesting to define the themes of teaching and devise ways to measure learning outcomes in distance learning, as compared to learning in a traditional classroom setting, based on the student experience. We could also expand the existing research work to include more advanced data (log) driven personalization.

References

Anzai, K., & Simon, H. A. (1979). The theory of learning by doing. *Psychological Review, 86*(2), 124–140.

Areias, C., & Mendes, A. (2007). A tool to help students to develop programming skills. *Proceedings of the 2007 international conference on Computer systems and technologies*, ACM.

Bati, T. B., Gelderblom, H., & Biljon, J. (2014). A

blended learning approach for teaching computer programming: design for large classes in Sub-Saharan Africa. *Computer Science Education, 24*(1), 71-99.

Baumberger-Henry, M. (2005). Cooperative learning and case study: Does the combination improve students' perception of problem-solving and decision making skills. *Nurse Education Today, 25*(3), 238-246.

Bayman, P., & Mayer, R. E. (1983). A diagnosis of beginning programmers' misconceptions of basic programming statements. *Communications of the ACM, 26*(9), 677-679.

Blondy, L. C. (2007). Evaluation and application of andragogical assumptions to the adult online learning environment. *Journal of Interactive Online Learning, 6*(2), 116-130.

Bonar, J., & Soloway, E. (1989). Preprogramming knowledge: A major source of misconceptions in novice programmers. In E. Soloway & J. C. Spohrer (Eds.), *Studying the novice programmer* (pp. 324-353). Hillsdale, NJ: Lawrence Erlbaum.

Brusilovsky, P., Sosnovsky, S., Lee, D., Yudelson, M., Zadorozhny, V., & Zhou, X. (2010). Learning SQL programming with interactive tools: From integration to personalization. *ACM Transactions on Computing Education, 9*(4), 19, pp. 1-15.

Chan, D. S. K. (2003). Validation of the clinical learning environment inventory. *Western Journal of Nursing Research, 25*(5), 519-532.

Clarke, I., Flaherty, T. B., & Mottner, S. (2001). Student perceptions of educational technology tools. *Journal of Marketing Education, 23*(3), 169-177.

Csikszentmihalyi, M. (1991). *Flow: The psychology of optimal experience*. New York, NY: Harper Collins.

Dacko, S. G. (2001). Narrowing skill development gaps in marketing and MBA programs: The role of innovative technologies for distance learning. *Journal Marketing Education, 23*(3), 228-239.

Détienne, F., & Soloway, E. (1990). An empirically-derived control structure for the process of program understanding. *International Journal of Man-Machine Studies*, *33*(3), 323-342.

Doymus, K. (2007). Effects of a cooperative learning strategy on teaching and learning phases of matter and one-component phase diagrams. *Journal of Chemical Education, 84*(11), 1857-1860.

Falkner, K., & Falkner, N. J. G. (2012) Supporting and structuring "contributing student pedagogy" in computer science curricula. *Computer Science Education, 22*(4), 413-443.

Fernandez, A., & Sanchez, J. M. (2003). CGRAPHIC: Educational software for learning the foundations of programming. *Computer Applications in Engineering Education, 11*(4), 167-178.

Fincher, S., & Petre, M. (2004). *Computer science education research*. London, UK: Taylor & Francis Group.

Helminen, J., Ihantola, P., Karavirta, V., & Malmi, L. (2012). How do students solve parsons programming problems: An analysis of interaction traces. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research* (ICER '12) (pp. 119-126). New York, NY: ACM.

Herrington, J., Oliver, R., & Reeves, T. C. (2003). Patterns of engagement in authentic online learning environments. *Australian Journal of Educational Technology, 19*(1), 59-71.

Hopkins, D. J., & King, G. (2010). A method of automated nonparametric content analysis for social science. *American Journal of Political Science, 54*(1), 229-247.

Hopson, M. H., Simms, R. L., & Knezek, G. A. (2001). Using a technology-enriched environment to improve high-order thinking skills. *Journal of Research on Technology in Education, 34*(2), 109-119.

Hundhausen, C. D. (2002). Integrating algorithm visualization technology into an undergraduate algorithms course: Ethnographic studies of a social constructivist approach. *Computers & Education, 39*(3), 237–260.

Kahney, H. (1983). What do novice programmers know about recursion? *In Proceedings from CHI'83: Human Themes in Computing Systems* (pp. 235-239). New York, NY: ACM.

Koka, A., & Hein, V. (2003). Perceptions of teacher's feedback and learning environment as predictors of intrinsic motivation in physical education. *Psychology of Sport and Exercise, 4*, 333-346.

Lee, C. D., & Kahnweiler, W. M. (2008). The effect of mastery learning technique on the performance of a transfer of training task. *Performance Improvement Quarterly, 13*(3), 125-139.

Lewis, C. M. (2010). How programming environment shapes perception, learning and goals: Logo vs. scratch. In *Proceedings of the 41st ACM technical SYMPOSIUM on Computer SCIENCE EDUCATion* (pp.346-350), ACM.

Lister, R., Berglund, A., Clear, T., Bergin, J., Garvin-Doxas, K., Hanks, B., Hitchner, L., Luxton-Reilly, A., Sanders, K., Schulte, C., & Whalley, J. L. (2006). Research perspectives on the objects-early debate. *SIGCSE Bulletin*, *38*(4), 146–165.

Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the Fourth International Workshop on Computing Education Research, ICER '08* (pp. 101–112), ACM.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004). Scratch: A sneak preview. In *Proceeding of the Second International Conference on Creating, Connecting and Collaborating through Computing* (pp. 104-109), IEEE.

Mayer, R. E. (2003). The Promise of multimedia learning: Using the same instructional design methods across different media. *Learning and Instruction, 13*, 125-139.

Patriarcheas, K., & Xenos, M. (2009, August). Asynchronous distance education forum-Brainstorming vs. Snowballing: a case study for teaching in programming didactics. In *International Conference on Web-Based Learning* (pp. 322-331). Springer Berlin Heidelberg.

Prince, M. J., & Felder, R. M. (2006). Inductive teaching and teaching methods: Definitions, comparisons and research bases. *Journal of Engineering Education, 95*(2), 123-138.

Reese-Durham, N. (2005). Peer evaluation as an active learning technique. *Journal of Instructional Psychology, 32*(4), 338-343.

Sadler-Smith, E., Down, S., & Lean, J. (2000). Modern teaching methods: Rhetoric and reality. *Personnel Review, 29*(4), 474-490.

Saeed, N., Yang, Y., & Sinnappan, S. (2009). Emerging web technologies in higher education: a case of incorporating blog, podcasts and social bookmarks in a web programming course based on students' learning styles and technology preferences. *Educational Technology & Society, 12*(4), 98-109.

Saitta, E. K. H., Gittings, M. J., & Geiger, C. (2011). Learning dimensional analysis through collaboratively working with manipulatives. *Journal of Chemical Education, 88*(7), 910-915.

Salinger, S., Plonka, L., & Prechelt, L. (2008). A coding scheme development methodology using grounded theory for qualitative analysis of pair programming. *Human Technology, 4*(1), 9-25.

Simon (2009). A note on code-explaining examination questions. In A. Pears & C. Schulte (Eds.), *Proceedings of The 9th Koli Calling International Conference on Computing Education Research* (pp. 21–30), Koli, Finland.

Teague, D., Corney, M., Ahadi, A., & Lister, R. (2012). Swapping as the "hello world" of relational reasoning: replications, reflections and extensions. In M. de Raadt & A. Carbone (Eds.), *Proceedings of the 14th Australasian Conference on Computing Education (ACE '12),* (pp. 87–93) Australian Computer Society, Inc.

Thomas, L., Ratcliffe, M., Woodbury, J., & Jarman, E. (2002). Learning styles and performance in the introductory programming sequence. In *Proceeding of the 33rd SIGCSE Technical Symposium on Computer Science Education*, (pp. 33-37), Australian Computer Society, Inc..

Thompson, E. (2008). *How do they understand? Practitioner perceptions of an object-oriented program* (Unpublished doctoral thesis). Massey University, Auckland, NZ.

Thota, N., & Whitfield, R. (2010) Holistic approach to learning and teaching introductory object-oriented programming. *Computer Science Education, 20*(2), 103-127.

Whalley, J. L., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P. K. A., & Prasad, C. (2006). An Australasian study of reading and comprehension skills in novice programmers, using the bloom and solo taxonomies. In *Proceedings of the 8th Australasian Conference on Computing Education* (pp. 243–252), Australian Computer Society.

Wolf, C. (2002). Towards an interactive web-based adaptive learning environment to address individual learning styles. *European Journal of Open, Distance and E-learning,* 1-14. Retrieved from http://www.eurodl.org/materials/contrib/2002/2HTML/iWeaver.pdf

Zuckerman, O., Arida, S., & Resnick, M. (2005). Extending tangible interfaces for education: digital Montessori-inspired manipulatives. In *Proceedings of the SIGCHI Conference on Human themes in Computing Systems* (pp. 859-868).

BELLE SELENE XIA is a Researcher at the Department of Information and Computer Science at the Aalto University School of Science. Previously she was a Project Manager for the Software Business and Engineering Laboratory at the Department of Computer Science and Engineering in the Aalto University School of Science and Technology. Her education includes a Bachelors in Science and Technology in Computer Science from the Helsinki University of Technology, as well as a Masters in Science and Technology from the same university. She has collected a wide range of international working experiences from companies such as HiQ International, Helsinki Op Bank, Euroclear PLC, Asahi Kasei Corporation, and Louis Vuitton. Additionally, she has worked for Research International Finland, which is a prominent research partner of customer behavior in Finland. Her current research concentrates on a financial economics project at the University of Ghent in Belgium.

Acknowledgements

The author is grateful for the key comments presented by Prof. Dr. Lauri Malmi and distinguished researcher Dr. Päivi Kinnunen, as well as their valuable contribution to this study. The author would like to thank Florilla Consulting Company for funding of this project. The author is also grateful to the anonymous reviewers for their helpful and constructive comments.