DEVELOPING A TECHNOLOGY ENHANCED CS0 COURSE FOR ENGINEERING STUDENTS

Erno Lokkila[1], Erkki Kaila[1], Rolf Lindén[1], Mikko-Jussi Laakso[2] and Erkki Sutinen[2]
[1]*University of Turku Graduate School*
[2]*University of Turku*

ABSTRACT

The CS0 course in the curriculum typically has the role of introducing students into basic concepts and terminology of computer science. Hence, it is used to form a base on which the subsequent programming courses can build on. However, much of the effort to build better methodologies for courses is spent on introductory programming courses instead of the earlier course. In this article we present an experiment where a CS0 course at our university was redesigned to utilize educational technology and automatic assessment. The redesign was based on a collaborative education platform called ViLLE. New automatically assessed exercise types with immediate feedback were designed and utilized with already existing ones to cover all topics taught in the course. In the paper, we present the design principles and the implementation of the electronic material for the course as well as our experiences on adapting the technology in the course. A detailed description of different exercises and other tasks are also provided. Finally, we present results and statistics collected from the course implementation.

KEYWORDS

Active learning, Course redesign, Automatic assessment

1. INTRODUCTION

In the modern world, the quest for effectiveness has driven educational institutes to reform their teaching curricula. The department of Information Technology at the University of Turku underwent a curriculum reform, in which courses were redesigned to include a more active take on learning. Some courses were refactored heavily, while others only received a slight makeover. Formerly, the introductory courses consisted of a set number of lectures and a final exam, on which the students were graded. The main reason for only a slight makeover was one from the list by Bonwell and Sutherland (1996): limited resources. The introductory courses remained lecture-heavy. However, electronic study materials were introduced to the course, which allowed students to put the theory taught during lectures into practice with little to no additional work load to the teaching staff.

This paper provides results on an application of active learning strategies to an otherwise traditionally lectured course. Student performance data was collected from six instances of the course, from 2009 to 2015. The first three instances were purely lectured. Students considered the course to be very difficult and grade averages were low. Active learning strategies were introduced to the course in 2012 in the form of automatically assessed electronic exercises and lecture questions. Student activity was measured by lecture attendance and scores from the electronic exercises.

2. RELATED WORK

The term “active learning” is used to mean a strategy of teaching, wherein the student is actively taking part in the learning process (Bonwell & Eison, 1991). Instead of passively sitting through a lecture, students interact with the subject matter in some way; either through discussions with one another and the teacher, or read or write about the subject matter in order to learn the content being taught (Candido et al., 2007). Bonwell and Sutherland (1996) propose a conceptual framework for active learning: a continuum from the simple task to the complex task. Neither side is ‘better’ than the other, instead, the educator must choose which part of the continuum is best suited for the course and students in question and subject matter being taught. The pedagogy behind active learning is heavily constructivist: students, by being immersed and engaged in learning, create meaningful information based on their interaction with the subject matter (Montero-Fleta et al., 2012).

Other course reforms to adopt a more active learning method have met with success. Kaila et al. (2015) reformed a CS1 course by swapping half of the lectures into tutorials, wherein students work in pairs to solve problems. Lecture questions that were meant to activate students were introduced to the remaining lectures. Goodhew and Bullough (2005) adopted a CDIO approach in their refactoring of a materials science and engineering course. They outlined a clear strategy on how active learning is put to use on their course. Steps to increase the engagement of students with the exercises have also been applied. More interactive exercises have been introduced to courses to increase student engagement with the subject matter. For programming courses, Parson’s problems (Parsons & Haden, 2006; Lopez et al. 2008; Helminen et al., 2012) have been found to be a useful addition

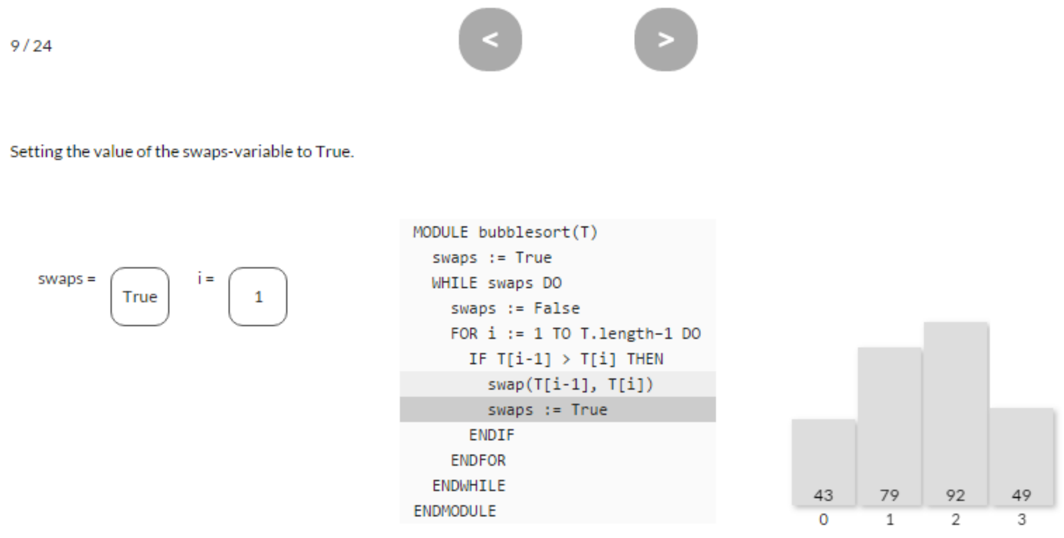
3. EXERCISES

At the heart of the refactoring are the exercises students were completing during the course. Instead of the traditional method of having students answer exercises on paper, we used the ViLLE learning platform (Laakso et al. 2016) to collect student answers. ViLLE provides students with immediate feedback on their performance in the given exercise. This is realized at the minimum by telling the students whether their answer is right or wrong and showing the correct answer when the given answer was incorrect. More elaborate feedback includes specific steps on how to solve the given problem, for instance, a quadratic equation. The reasoning behind favoring automatic assessment of exercises over manual assessment is to enable students to answer more exercises, and receive individual feedback on each answer immediately after answering.

ViLLE provides teachers with a multitude of different exercise types. The all-purpose multiple choice questions and fill-in exercises are available and were utilized on the course. Other, more specific exercise types used on the course were binary calculations, the number-base conversion exercise, an algorithm visualization exercise, programming exercises and Parson’s puzzles. Only the multiple choice and fill-in exercises were used early in the course, and the algorithmic and programming exercises were introduced after the half-point in the course, when more programming-oriented topics were discussed. Next the exercise types are explained in more detail.

The multiple choice questions and the Fill-in exercises were used to test a wide array of topics, including Signal-to-Noise ratios, number of bits transferred over networks and OSI-model. The ViLLE multiple choice question exercise allows for short open questions, in addition to the familiar question-and-several-choices format. As both these modes are automatically assessed, students are given feedback on the correctness of their answer immediately upon answering. Additionally, in case of an incorrect answer, the students were provided with an explanation why the given answer was incorrect. Moreover, the open-ended questions can be randomized to provide different numeric values for, example, file sizes or network bandwidths for each attempt. This is an effective method to ensure students have a basic understanding for what is being asked, as they cannot merely copy the answer since all students have effectively different question parameters.

The algorithm visualization exercise is implemented using the JavaScript Algorithm Visualization (JSAV) library (Karavirta& Shaffer 2013) and was used to visualize execution of various sorting algorithms. The algorithm was visualized with pop-up questions at key points of the algorithm, all questions were automatically assessed and feedback was given to the students on where they went wrong. Because students only had a very cursory knowledge of programming concepts, visualizing the execution of the different algorithms was meant to not only teach them the workings of said algorithm, but enforce the concepts of looping and branching as well.



Setting the value of the swaps-variable to True.

```
MODULE bubblesort(T)
  swaps := True
  WHILE swaps DO
    swaps := False
    FOR i := 1 TO T.length-1 DO
      IF T[i-1] > T[i] THEN
        swap(T[i-1], T[i])
        swaps := True
      ENDIF
    ENDFOR
  ENDWHILE
ENDMODULE
```

Figure 1. JSAV visualization exercise

The binary calculations in ViLLE are an interactive version of calculating binary calculations column by column. The main benefits of this exercise in comparison to calculating on paper are the possibility to automatically assess the answers and the immediate feedback students receive. The binary calculations in ViLLE are randomly generated from parameters given by the teacher. Immediate feedback in the binary calculation exercise consists of showing the student the correct answer to the given problem.

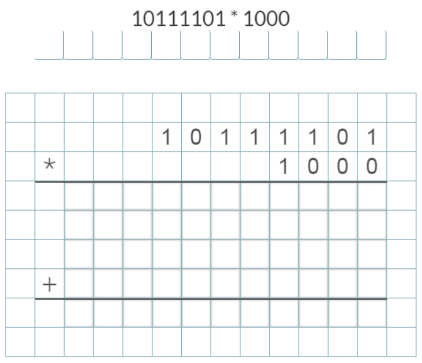
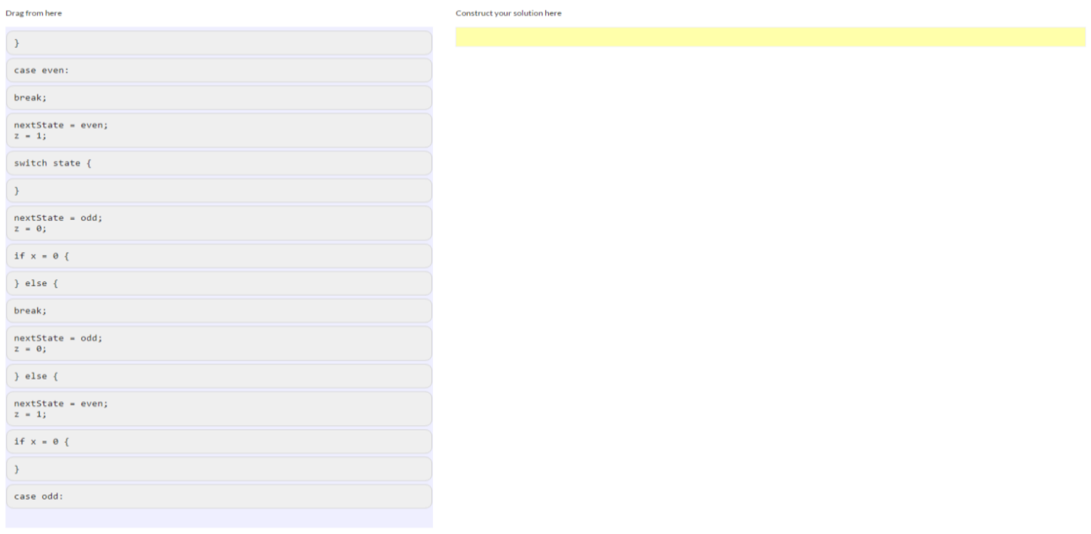


Figure 2. Binary calculations

Parsons' problems (Parsons 2006) are a programming exercise, where the student is given ready program code, but in an incorrect order (Fig). The given program code must then be ordered correctly, usually by dragging and dropping the rows of code to the correct order. Parsons' exercises have been found to involve a similar skill set as that of writing code (Denny, 2008). During this course, students were taught a cursory understanding of basic control structures and syntax, which means that students recognize loops and branching in code, but may not be able to write working programs in the same programming language. Parsons' problems were then used to practice algorithmic thinking and problem solving without the need to memorize and write actual program code.



```
}
case even:
break;
nextState = even;
z = 1;
switch state {
}
nextState = odd;
z = 0;
if x = 0 {
} else {
break;
nextState = odd;
z = 0;
} else {
nextState = even;
z = 1;
if x = 0 {
}
case odd:
```

Figure 3. Parsons' problem

4. REFACTORING THE COURSE EXAM

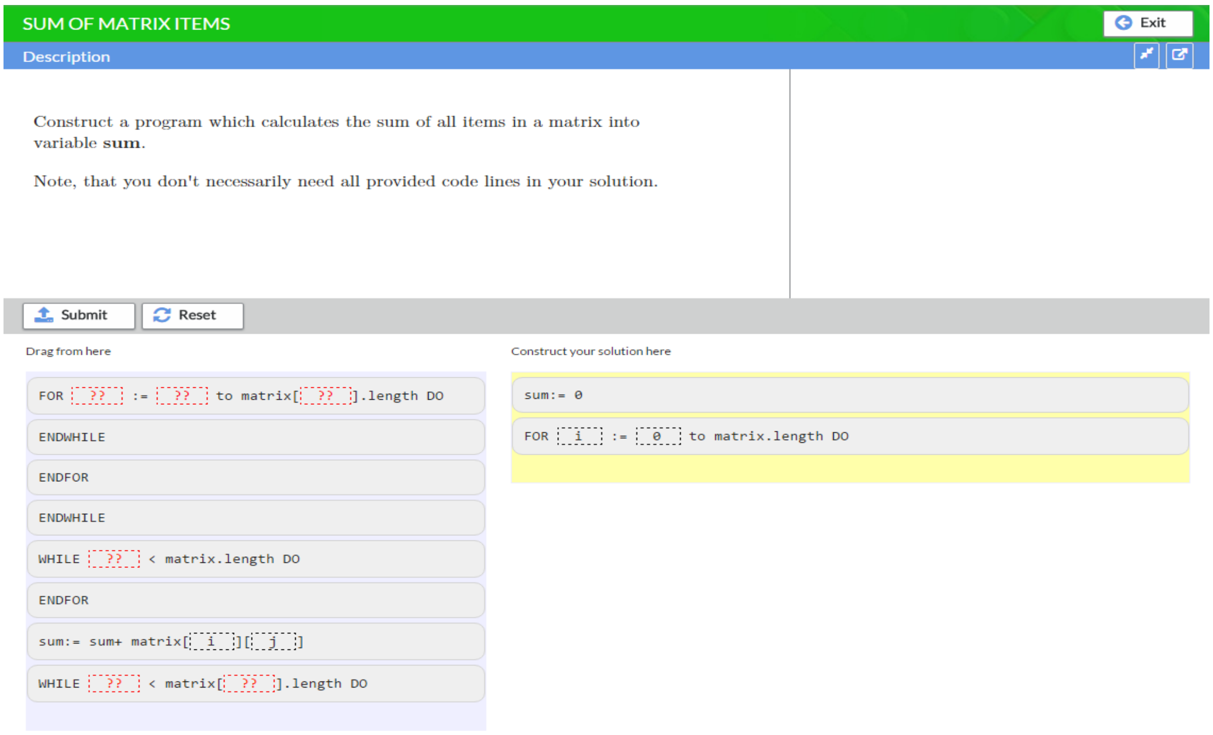
As part of the development, the course exam was decided to be organized in an electronic form by using ViLLE. There are several reasons to use electronic exam instead of pen and paper. First, by using a suitable tool, the answers can be automatically assessed, which drastically decreases the course staff's workload and quickens the evaluation process. Second, using an electronic exam enables the usage of authentic coding (or code constructing) exercises with possibilities for compiling, testing and debugging the code before submission. Thirdly, an electronic exam provides a possibility for randomizing the exercises and exercise parameters, which makes it possible to offer more heterogenic exams.

Since ViLLE was used comprehensively in the course for practicing the topics and for recording attendances, it was decided to be used as an exam platform as well. The students were already familiar with the system, which meant that students' cognitive load (see for example De Jong, 2010) was not unnecessarily increased in the final exam. The exercises were selected to resemble the ones practiced during the course. The exam structure is displayed in Table 1.

Table 1. The course final exam structure

Question #	Type	Description
1a, 1b and 1c	MCQ and open questions	Multiple choice and open questions about all topics in the course with emphasis on theory.
2a	MCQ	OSI Model: question about each level in the model
2b	Questions	Questions about processor technology and other technology
3a and 3b	Conversion	Binary / decimal / hexadecimal conversions between all formats
3c	Logical operations	Logical operations using bit patterns
3d and 3e	Questions	Calculations and format conversions
3f	Questions	Questions about SNL ratio and sine signal
3g	Calculations	Huffman coding
3h	Calculations	FIR, DFT and FFT algorithm
4a	Algorithm visualization	Simulating a bubble sort or insertion sort algorithm
4b and 4c	Code construction	Building a pseudo code algorithm using given code lines
4b and 4c (alternatively)	Coding	Writing a program according to given specifications using either Python or C

Most of the questions are either initialized with random parameters, or contain a pool with five or more alternatives of questions. One of these questions is randomly assigned to each answerer. The questions are varied between instances, but the basic principles stay quite similar. Since 2013 instance, the students could have chosen either a code construction or a coding exercise in the two final tasks, and in the final instance (2015) construction exercises were provided solely instead of coding exercises. This was mainly done because programming is taught in other introductory courses. An example of a code construction exercise is displayed in Figure 4.



Construct a program which calculates the sum of all items in a matrix into variable **sum**.

Note, that you don't necessarily need all provided code lines in your solution.

```
FOR ?? := ?? to matrix[ ?? ].length DO
ENDWHILE
ENDFOR
ENDWHILE
WHILE ?? < matrix.length DO
ENDFOR
sum:= sum+ matrix[ i ][ j ]
WHILE ?? < matrix[ ?? ].length DO
```

```
sum:= 0
FOR i := 0 to matrix.length DO
```

Figure 4. Code constructing exercise in ViLLE's exam mode. Note, that some of the code lines are parameterized

Although the electronic exams in our university are usually supervised, the exam for the CS 0 course was decided to be organized as unsupervised. This meant, that the students could take the exam wherever they wanted by using their own computers. A possibility to take the exam in the computer lab was still provided for students who could not use their own computers. The exam start and end times were still restricted, meaning that all students took the exam at the same time.

Since most of the questions were randomized, the supervision was not deemed necessary. It is likely, that some students provided assistance to other students during the exam, but because of randomization and question pools it is highly unlikely that two students taking the exam at the same premises answered to more few same questions. Moreover, we wanted the students to be able to use internet and other necessary resources to help them solve the tasks, as this is usually the case when they are applying the skills into real-life problems later.

5. RESEARCH SETUP

The course was designed and utilized at 2012. The results are observed from four instances (2012 to 2015). The number of participants (excluding the students from other departments) in the instances are displayed in Table 2.

Table 2. Instances of the course following the new design

Year	**2012**	**2013**	**2014**	**2015**
N	71	61	46	29*[1]

The significant decrease in number of students taking the course between instances of 2013 and 2014 was due to changes in the department's curriculum. While in 2012 and 2013 both, computer science majors and the engineering students (with IT major) took the course (along with other students from the faculty), a new specialized course was designed as a replacement for CS majors in 2014.

For comparison, results of the old course instances between 2009 and 2011 are also presented, though the major changes in course content, method and personnel mean that not fully valid comparisons can be made; this is also further addressed in the Discussion section. The number of participants in instances of 2009 to 2011 is displayed in Table 3.

Table 3. Instances of the course with the old design

Year	**2009**	**2010**	**2011**
N	77	83	58

In addition to grade averages of the old and redesigned course instances, the ViLLE scores obtained as well as the average number of attendances are also displayed. ViLLE automatically collects a huge amount of data about the submissions made, including for example the scores, submission times and the time spent on each individual task.

6. RESULTS

The grade averages (in scale of 1 to 5, where 5 is the best) of all course instances are displayed in Table 4.

Table 4. The grade averages of all course instances following the new design

Year	**2012**	**2013**	**2014**	**2015**
Grade average	4,15	4,54	3,78	3,59*[1]

As seen in the table, there is a slight drop in the course average between instances of 2013 and 2014. The main difference between years 2013 and 2014 is the decrease in number of students taking the course, as a separate course for computer science majors was started at 2014. Hence, there were only engineering students with information technology major at instances of 2014 and 2015. The content of the course was change at 2014 accordingly.

For comparison, the grade averages for old course instances are also displayed in Table 5.

Table 5. The grade averages of all course instances following the old or new design

Course	Old	Old	Old	New	New	New	New
Year	2009	2010	2011	2012	2013	2014	2015
Grade average	3,55	3,05	3,12	4,15	4,54	3,78	3,59

As seen in the table, the students' grade averages seem to be higher in the new course. This can be confirmed by calculating the averages of the old course instances (total N = 217, average = 3.2569) and the new instances (N = 207, average = 4.10628). The difference is statistically significant, with $p < 0.01$ (calculated with Wilcoxon non-parameterized rank-sum test). However, as stated before (and readdressed in

*[1] The data for 2015 was not conclusive when this article was written, as there were still revision exams to be held for the course.

the discussion), the comparison of the courses is not fully valid because of changes in the content, methodology and course staff.

The student activity in ViLLE was compared to grade obtained in each instance. The results are displayed in Figure 5.

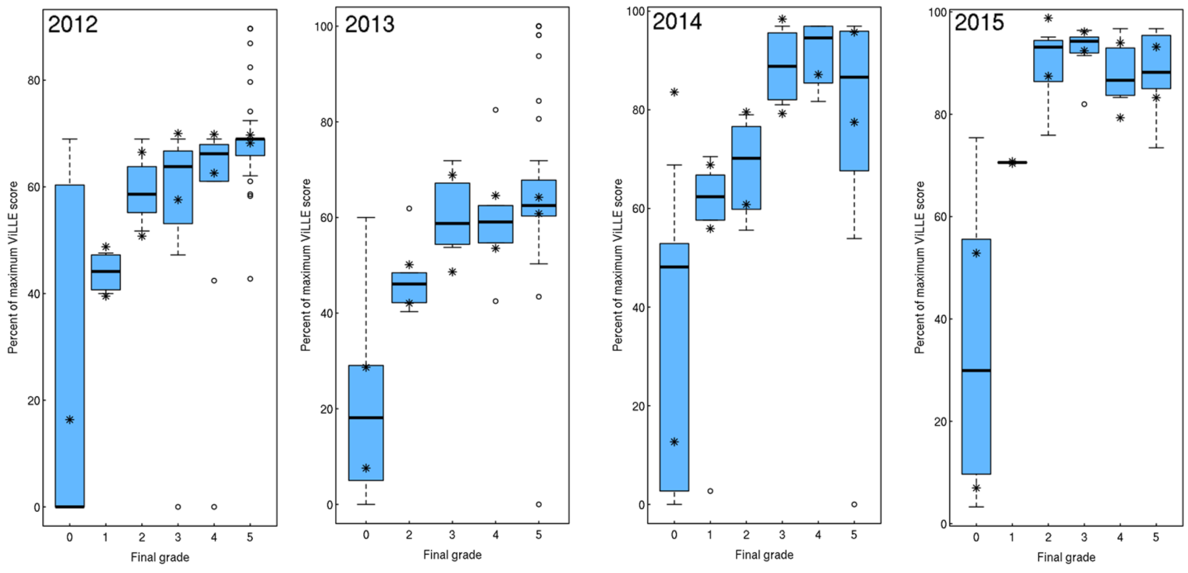


Figure 5. The relationship between course grades and the amount of ViLLE exercises completed during the course

In each of the images in the figure, the final grade is displayed in the X-axis and the percentage of ViLLE exercises done in the Y-axis. As seen in the figure, at each instance the students who worked the hardest during the course also got the higher grades.

The attendances in the course lectures were recorder by using RFID devices and RFID tags or cards given to students. ViLLE automatically registered an attendance when a tag was used with a reader in a lecture hall. The average number of attendances is displayed in Table 6.

Table 6. The average amount of attendances in the course lectures

Year	**2012**	**2013**	**2014**	**2015**
Attendance avg.	5,05 of 7	4,32 of 6	5,98 of 7	5,25 of 7
% of maximum	72,14%	61,71%	85,43%	75%

As seen in the table, the attendance rates were higher at the latter instances (2014 and 2015) when the attendees were engineering students only. Hence, it seems that the major students are more likely to attend the lectures than others

7. DISCUSSION

It seems that the design of the new course was successful. The thorough utilization of educational technology and the new exercise types designed encouraged students to work quite hard. Moreover, it seems that the students who answered to most exercises also got the highest grades. There are some differences between the instances of the new course, for example the grade average decreased a little between instances of 2013 and 2014. There are a few likely explanations for this: first, the course content was altered between the aforementioned instances to better fulfil the specific demands of our engineering education. At the earlier instances the content needed to be suitable for students with other majors as well. Moreover, the data for the latest instance was not comprehensive when this article was written, as there were students who had not taken the exam yet (and some students who were likely to retake the exam to improve their grade), so the grade average for 2015 is likely to increase from the currently reported.

The students also seemed to participate into lectures quite actively. Notably, when the course consisted of engineering students only (instances 2014 and 2015) the average number of attendances was higher than in the earlier instances. It is likely, that the engineering students find the topic more interesting and more relevant to their other studies, and hence attend the lectures more than students with other majors (such as math or physics majors).

REFERENCES

Bonwell, C. C., & Eison, J. A. (1991). *Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports*. ERIC Clearinghouse on Higher Education, The George Washington University, One Dupont Circle, Suite 630, Washington, DC 20036-1183.

Bonwell, C. C., & Sutherland, T. E. (1996). The active learning continuum: Choosing activities to engage students in the classroom. *New directions for teaching and learning*, *1996*(67), 3-16.

Candido, J. P., Murman, E. M., & McManus, H. (2007). Active Learning Strategies for Teaching Lean Thinking.

De Jong, T. (2010). Cognitive load theory, educational research, and instructional design: some food for thought. *Instructional Science*, *38*(2), 105-134.

Denny, P., Luxton-Reilly, A., & Simon, B. (2008, September). Evaluating a new exam question: Parsons problems. In *Proceedings of the fourth international workshop on computing education research* (pp. 113-124). ACM.

Goodhew, P, & Bullough, T. “Active learning in materials science and engineering”. Proceeding of the

1st Annual CDIO conference Queen’s University Kingston, Ontario, Canada, 2005

Helminen, Juha, et al. "How do students solve parsons programming problems?: an analysis of interaction traces." *Proceedings of the ninth annual international conference on International computing education research*. ACM, 2012.

Kaila, E., et al. "Comparing student performance between traditional and technologically enhanced programming course." *Proceedings of the Seventeenth Australasian Computing Education Conference*. 2015.

Karavirta, V., & Shaffer, C. A. (2013, July). JSAV: the JavaScript algorithm visualization library. In Proceedings of the 18th ACM conference on Innovation and technology in computer science education (pp. 159-164). ACM.

Laakso, M.-J., Kaila, E. & Rajala, T. 2016. ViLLE – Designing and Utilizing a Collaborative Education Tool. Sent to British Journal of Education Technology.

Lopez, Mike, et al. "Relationships between reading, tracing and writing skills in introductory programming." *Proceedings of the fourth international workshop on computing education research*. ACM, 2008.

Montero-Fleta, B. (2012). Looking beyond linguistic outcomes: active learning and professional competencies in higher education. *Procedia-Social and Behavioral Sciences*, *46*, 1812-1819.

Parsons, D, & Patricia Haden. "Parson's programming puzzles: a fun and effective learning tool for first programming courses." *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52.* Australian Computer Society, Inc., 2006.