A Model for Gamifying Programming Education: University-Level Programming Course Quantified

E. Kaila*, M.-J. Laakso*, T. Rajala* and E.Kurvinen*
* Department of Future Technologies, University of Turku, Finland
{ertaka, milaak, temira, emakur} @utu.fi

***Abstract* - Utilizing gamification in course holistically requires that all areas of the course can be quantified, and the progress made transparent to students as well. However, keeping track of all students' scores, submissions and other tasks can be too big of a workload to be practical, especially in the larger courses. We have successfully adapted a collaborative education tool to university-level programming courses. The tool is used to record student performance in all areas of the course, including for example attendances, demonstration and tutorial scores, weekly surveys and additional exercises. In addition to providing comprehensive statistics to teachers, all progress is also visible to students in real time. In this paper, we describe the re-design of the course, with gamification and transparency to students in focus. Moreover, we analyze students' scores in different components of the course, and find out if there are correlations to be found between different areas. The results seem to indicate, that methods emphasizing active learning are the most beneficial for students' performance in the course.**

Programming education; gamification; quantification; student performance;

I. INTRODUCTION

Programming is an essential skill in computer science and information technology education. Hence, various studies have been conducted about how the introductory programming courses should be taught, and what kind of tools can be utilized to improve student performance. Typical changes in the course methodology include for example emphasizing active learning (typical application is Flipped learning, [1, 2]) and utilizing gamification [3]. However, often the effect of changes in the methodology is not evaluated in detail. Although holistic changes in the student performance, motivation or other factors are important, it is interesting (and also important) to find out how the students perform in courses' different components and what is their effect into the final performance.

We have previously presented a tutorial-based programming methodology [4, 5] which was applied to an introductory programming course. The emphasis of changes is in active learning, which is implemented with tutorials (combinations of learning material and automatically assessed exercises), additional exercises and demonstrations. Gamification is utilized by making all scores transparent to students, by utilizing virtual trophies and by offering external reward (in form of small bonus to course grade) to student for completing more than a required minimum.

In this article, we explore the students' performance during the course by quantifying different components. Moreover, we discuss the effect of individual components towards students' performance in the final exam. The paper is structured as follows: first, related research by the community is presented. Next, the redesigned course is discussed along with the educational technology used and the quantifying of different components in the course. Next, students' performance in different components (and in total) is visualized and the correlation of components with the final exam presented. Finally, the results are discussed and the article concluded with possibilities to future research opportunities..

II. RELATED WORK

Programming is a difficult task to learn. According to several studies (for example [6, 7]), typical issues are for example the lack of adequate mental models, using overly simplified strategies and too little time spent on planning and testing problems. Several solutions have been provided to fix this problem. Educational technology, for example, can potentially help students to perform better. Examples of such technology are program visualization [8, 9], where program visualization is made visible step by step, and the tools for automatically assessing program code [10, 11], which can significantly reduce the time spent on evaluating the programs written by students. However, the tools alone cannot solve the problems in the programming courses, more important is how they are used and how active learning is emphasized in courses.

Different methodologies have been utilized to try to improve the outcome and the students' experience in programming courses. Boyle et al. [12] report an experience where blended learning was utilized in introductory course. They found out, that the online environment used in the course (along with other changes) resulted into better pass rates. In a systematic review [13] about different methodologies used in programming education, the authors found that of all reviewed features, the most effective ones were the utilization of relatable content, collaboration and adapting an introductory course (typically referred as CS0) before the introductory programming course.

A review of gamification literature [14] identifies several gamified elements that can be used in education. These include for example points, levels or stages, progress bars and feedback. Gamification can potentially increase students' motivation in courses. In [15] the authors present an experiment where game thinking and game mechanics were utilized with a software development team. The authors found out, that almost 80 % of the students favored the introduced, gamified mechanics in comparison to traditional setups. Similarly, in [16] the authors found out, that utilizing gamification to enhance online student collaboration lead up to 88% more efficient collaboration. Moreover, [17] reports various motivational impacts in different study groups when gamification was utilized by digital badges.

III. About the Course

As an example course, the Basic Course of Algorithms and Programming is used. The course is taught annually at University of Turku, Finland, and serves as essential part of programming education offered for computer science majors and other students. Content-wise, the course is quite typical CS1 course. It is taught using Java, and focuses on procedural paradigm. The course lasts for seven weeks and is concluded with a final exam at the eight week.

The topics covered in seven weeks of the course are listed in Table 1.

TABLE 1. THE CONTENTS OF THE COURSE, DIVIDED INTO 7 WEEKS.

Week #	Topic
1	Introduction to course, transfer from Python to Java
2	Variables, strings, conditional statements
3	Repetition
4	Methods (i.e. procedures and functions)
5	Arrays
6	Using external modules, other data structures
7	Summary

As seen in the table, the course covers basic introductory concepts related to procedural programming. Each week there are two 2-hour sessions. Moreover, there are a total of four 2-hour demonstration sessions where students demonstrate and discuss their answers to programming tasks.

A. Utlizing Educational Technology

The implementation of the course relies heavily on usage of educational technology. Because of this, a suitable tool was required when the course was redesigned. ViLLE, an exercise-based collaborative education tool was chosen to be used in all areas of the course. ViLLE has a support for multiple programming languages (including Java, Python, C#, C++, JavaScript, to name a few). Moreover, there are more than one hundred different exercise types. A number of these types are designed especially for programming education, including for example program visualization [18], Parsons puzzles [19] and code writing. There is also a program simulation exercise, where the students need to simulate program code execution (for example by creating and modifying variables and method instances) while the example program is executed. All these exercise types are automatically assessed and provide immediate feedback upon submission. An example of programming-related ViLLE exercise is shown in Figure 1.

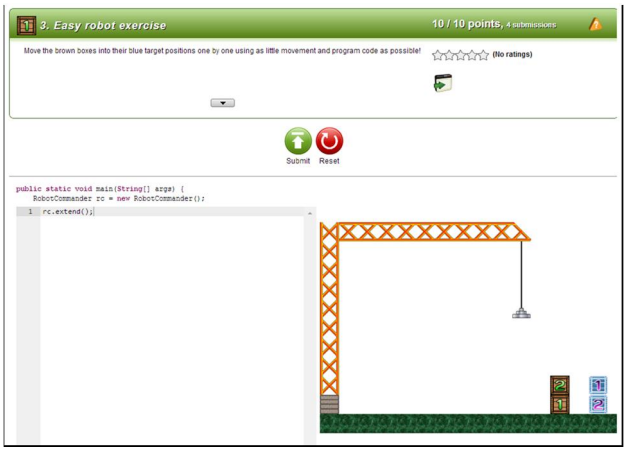

Figure 1. An example of programming-related exercise in ViLLE. The students need to move the boxes into their target positions by writing a program.

ViLLE works as a web-based application, usable with any modern browser with no additional plugins required. The exercises are collected into rounds, and a number of rounds form a course. There are three types of rounds needed in the programming course: normal rounds, exam rounds and tutorial rounds. All rounds have an opening and closing times set by the course teacher. Exam rounds offer a possibility to modify several properties, including the amount of feedback given. Tutorials in ViLLE are combinations of learning material (such as text, images, videos or code examples) with automatically assessed exercises. All exercises in ViLLE can be done in collaboration, but the group work is emphasized in tutorials. Both students (or all students, as the group size can be anything between 2 and 9) sit in front of same computer solving the exercises together, and the points are awarded to all students in the group.

ViLLE can also be used for recording other course activities. Student attendances can be collected automatically by using RFID (Radio-frequency identification) tags handed to students. When a tag is used with a reader in any classroom, the attendance is automatically registered. The same mechanism can be used to register demonstration points as well. ViLLE has a support for surveying students' perceptions, which can be a powerful tool in course quality control [20]. Survey answers and all other statistics can be easily viewed in ViLLE by course teachers. The statistics can also be exported as a plain text file or as a Microsoft Excel spreadsheet for further analysis. Comprehensive description of the tool can be found in [21].

B. Course redesign

With the redesign, several design principles were chosen [5]. The redesigned course emphasized active learning by replacing half of the lectures with tutorials.

Moreover, the remaining lectures were accompanied with additional ViLLE exercises which were meant to be answered right after the lecture. To enable student feedback, a simple survey was used after each lecture and tutorial. The survey consisted of three questions: "What did you learn this week?", "What things remain unclear after this session?" and "How would you improve this session?". The answers to these questions were analyzed after each session, and the consecutive sessions were modified, if necessary. Finally, the course final exam was transferred into electronic exam. Utilizing electronic exam enables the students to test, modify and debug their answers, bringing the whole process a lot closer to real programming. Combining electronic exam with automatic assessment also reduces teachers' workload significantly and can make the assessment process more reliable. In the redesign process, three non-affiliated researchers confirmed that the electronic version is at least as difficult as the traditional one.

The structure of the redesigned course and a comparison to course instance before the redesign process is displayed in Table 2.

TABLE 2. COURSE INSTANCES COMPARED.

Component	Old instance	New instance
Lectures	7 x 2 x 2h	7 x 2h
Tutorials	None	7 x 2h
Demonstrations	4 x 2h	4 x 2h
ViLLE exercises	None	3 to 5 weekly
Exam	Pen and paper	Electronic
Exam tasks	2 to 3	8 to 9
Attendances	Not registered	Registered

There is not a significant difference in students' work load when measured in hours in a week. However, in the new instance, the amount of active learning via different kinds of exercises is a lot larger than in the old version.

C. *Quantifying the Course and Utilizing Gamification*

As seen in Table 2, the new course instance consists of five components: lectures, tutorials, demonstrations, ViLLE exercises and the final exam. The first four components are split among several weeks of the course. A maximum score for all areas was defined to give them proper weight in the whole evaluation scheme. The scores are displayed in Table 2.

TABLE 3. THE MAXIMUM SCORES AND REQUIRED MINIMUMS FOR DIFFERENT COURSE COMPONENTS.

Component	Max. score	Required minimum
Lecture attendance	200	None
Tutorials	650	325 (50 %)
Demonstrations	400	200 (50 %)
ViLLE exercises	200	100 (50 %)

Hence, the total maximum score at the course becomes 1450. The final exam was scored in scale of 0 to 90, but this score was independent of the course total score. Exceeding the minimum limits gave student a small bonus into the final grade. However, this bonus was only applied if at least 50 % of the points was collected from the final exam.

To enable the gamifying effect from collecting points in different categories (and in total), the progress must be made visible to students. Hence, in ViLLE the students can see their real-time total score, round scores and the scores of individual assignments all time. The effect can be boosted by using virtual trophies: hence, a bronze trophy is awarded from reaching minimum score in a round (50 %), silver trophy for 75 %, gold trophy for 90 % and the diamond trophy for the full score. An example displaying the trophies and the score in ViLLE is displayed in Figure 2.

Figure 2. Students' score displayed with a progress bar and virtual trophies.

The figure displays one round with 6 exercises. The user has completed the first five exercises with full points but has not started the final one yet. The bar in the top displays the full score gathered from the round (50 out of 60) and the trophies already reached (bronze and silver).

D. *Earlier Results*

In [5] we studied the effects of course redesign on student performance on the course. An old instance of the course (N=210) was compared to an instance using the redesigned methodology (N=193). We found out that the pass rate increased statistically significantly in the new instance. Curiously, the course average grade remained almost the same. A summary of the results is collected into Table 4.

TABLE 4. THE RESULTS OF THE OLD METHODOLOGY AND NEW METHODOLOGY COMPARED.

	Old methodology	New methodology
N	210	193
% passed course	53.33 %	80.82 %
Grade mean (1 to 5, 5 best; of accepted)	3.63	3.57

As seen in the table, performance-wise the redesign was highly successful. The pass rate in the new version of the course is significantly better. Next, we quantify the students' performance in different areas of the course, to try to find out how the better pass rate was achieved.

IV. RESEARCH

ViLLE automatically collects a lot of data from every submission made by students. For example, submission score and the time used answering are required. Additionally, exercise-related data is stored. A coding exercise, for example, stores all code student has written for the submission. This opens up new possibilities for studying student misconceptions and coding style [22]. In this paper, the focus is in quantifying students'

performance, and hence on the scores collected. A single instance of the course is studied. For consistency, the same instance is observed than in the previous study [5] about students' performance.

A. *Results*

Students' total scores are visualized in Figure 3.

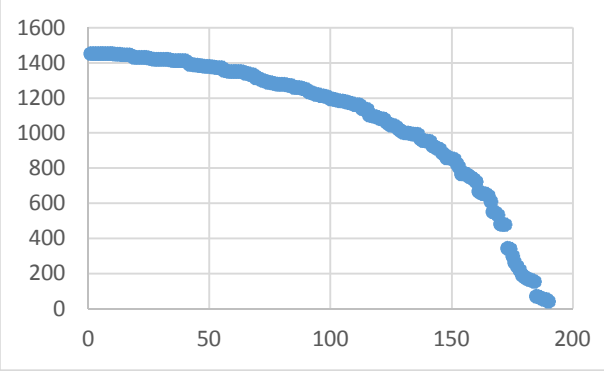

Figure 3. Students' total score collected from the course visualized.

As seen in the figure, majority of students collected points quite well. The average number of points collected was 1077.0 out of 1450. The distribution of total points between different course components is visualized in Figure 4.

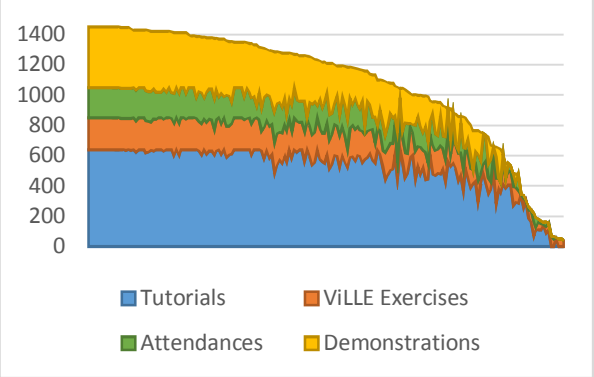

Figure 4. All students' score collected from the course categorized into different components.

As seen in the table, the majority of collected points comes from tutorials and the demonstrations. This is in line with the intended workload of the components.

The course lasted for seven weeks (with exam at the eight week). The average points collected from the tutorials, ViLLE exercises and attendances each week are displayed in Figure 5.

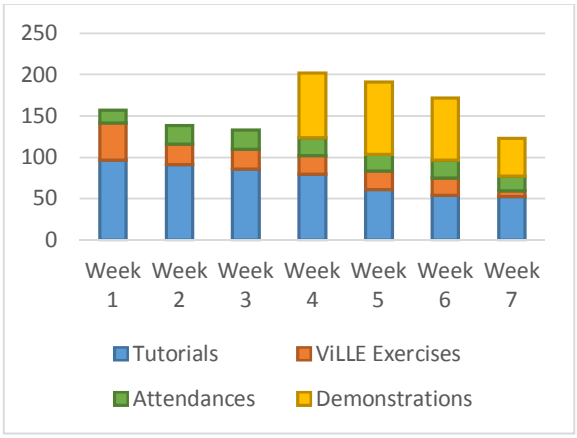

Figure 5 points collected from all components of the course weekly.

There are some important points to notice about the figure. First, the maximum amount of ViLLE exercises was 50 at the first week, and only 10 at the last week. For weeks 2 to 5 the maximum was 30 points. This explains the larger average on the first week and the smaller one on the last week. Second, attending the first lecture only provided students 10 points, while all the remaining ones awarded 15 points. Hence, the average number of points collected from ViLLE exercises and attendances is quite constant during the whole course. However, the number of points collected from the tutorials drops noticeably in the second half of the course. This is very likely due to topics getting more difficult when the course advanced. There were four demonstrations, taking place in weeks 4 to 7. The amount of points collected seems to drop a little towards the end (the average for last week was 45.7 while in the first three weeks the average was more than 75 points). Again, the last set may have been more difficult than the previous ones.

To find out whether the work done had an effect to the course grade, a correlation between total points collected and the final exam points (without bonus) was calculated. The mapping of students is shown in Figure 6.

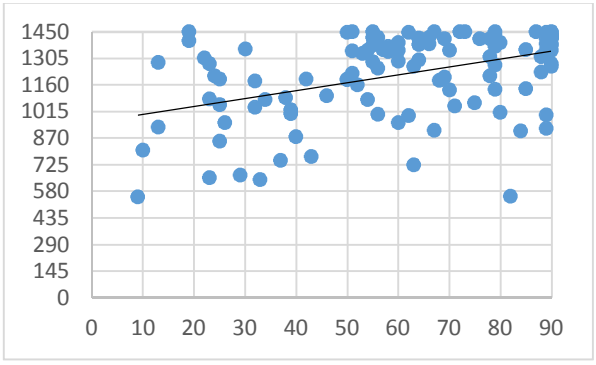

Figure 6. Students plotted according to their total score collected (y-axis) and the point in final exam (x-axis) without added bonus.

The figure is by no means ambiguous. Still, it seems that the majority of students who collected a lot of points during the course also did quite well in the final exam. As there were minimum requirements for collecting points from different components, the students who did not meet them were not allowed to participate into the final exam. Hence the bottom half of the figure is mostly empty. A

correlation value of r = 0.443 between the total points collected and the final exam score does indicate a moderate correlation – a fact that is also visible in the figure. The exam score correlation with all course components is displayed in Table 5.

TABLE 5. CORRELATION BETWEEN COURSE COMPONENTS AND THE FINAL EXAM SCORE WITHOUT ADDED BONUS.

Component	Correlation with exam score (r)
Total score	0,443657
Tutorials	0,495501
Attendance	0,156899
ViLLE exercises	0,345411
Demonstrations	0,39234

As seen in the table, the tutorials have the highest correlation with the final exam score. Hence, the emphasis of tutorials in the course total score seems to be justified. Curiously, lecture attendances only have a minor correlation (0.157) with the points collected from the final exam. Hence, active learning methods seem to be the most beneficial for learning.

V. DISCUSSION

The students seem to have collected points quite well from all course components. The average amount of 1077.0 (out of 1450) is rather high average score, especially since the students who dropped out from the course (pass rate was 80.82 %) are included in the number. Most of the points collected came from tutorials and demonstrations. This emphasizes the course's focus on active learning. Curiously, the amount of points collected from these topics also decreased towards the end of the course. We can think about two possible explanations for this. First, it is possible that the exercises got more difficult towards the end of the course, and the students were not able to complete as many of them as in the beginning. Second, it is also possible that the students were not as eager to collect points when they had completed the required minimum amounts from all components. However, the average amount of points collected from ViLLE exercises and attendances remained constant during the course – likely because the points were easier to collect from these categories.

There is a moderate correlation (r = 0.443) between the total score collected and the final exam score (without the bonus points collected during the course). This indicates, that although there is a correlation between the work done during the course and the skill level in the final exam, there are some exceptions that need to be considered. Collaboration has been proven quite useful to programming learning [23, 24, 25, 26]. However, there is a possible downside to doing tutorials and demonstrations collaboratively: although the students are instructed to work together and change the operator (i.e. the student who is using the keyboard and mouse) frequently, it is still possible that some students may have adopted a passive role. Similarly, it is possible to copy the demonstration answers from other students. There are also some students who did quite well in the final exam without completing much points during the course. These are probably students with previous programming knowledge.

Of all course components, tutorials had the strongest correlation with the final exam score. Moreover, lectures had the smallest correlation with the exam. Hence, it seems that active learning methods are the most useful for students' performance. Although not explicitly researched in this study, we think that making scores of all components transparent to students is important. Displaying score (with possible virtual trophies) and offering a reward for completing more than the required minimum are important, gamifying factors that may have a positive effect towards student motivation.

VI. CONCLUSION AND FUTURE WORK

Quantifying the individual score components seems to provide valuable information for teachers and education researchers. Students' interest in completing different areas can potentially be increased by utilizing gamification by making the scores transparent and providing rewards on completing more than required work. In the future, the effect of gamification should be observed better. There are some possibilities to complete this: first, we could have course instances that are similar by all other aspects, but the other would provide gamified features (such as ones utilized in the course described here). Second, it may be useful to survey students for their perceptions about gamification in learning context.

Although there was only moderate correlation between the total score collected in the course and the course final exam scores, it still seems that the students benefit from learning actively with versatile methods. In the future, it would be useful to observer the correlations and specific effects of individual components in the course even closer. This way it would become possible to emphasize the most effective methods. Still, utilizing different kinds of methods in the course provides students heterogeneous ways to learn the topic, and can hence increase motivation and make learning programming more interesting. Finally, yet another direction to future research is to quantify the effect of components in different kinds of courses to find out if the effects can be generalized.

REFERENCES

[1] A. Sams and J. Bergmann, "Flip your students' learning. Educational Leadership", vol. 70(6), 2013, pp.16-20.

[2] A. Amresh, A. R. Carberry and J. Femiani, "Evaluating the effectiveness of flipped classrooms for teaching CS1". In 2013 IEEE Frontiers in Education Conference (FIE), 2013, pp. 733-735

[3] S. Deterding, D. Dixon, R. Khaled and L. Nacke, "From game design elements to gamefulness: defining gamification." In Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, ACM, 2011, pp. 9-15

[4] E. Kaila, T. Rajala, M.-J. Laakso, R. Lindén, E. Kurvinen, V. Karavirta and T. Salakoski, "Comparing student performance between traditional and technologically enhanced programming course". In proceedings of the Seventeenth Australasian Computing Education Conference (ACE2015), Sidney, Australia, 2015.

[5] E. Kaila, E. Kurvinen, E. Lokkila, and M.-J. Laakso, “Redesigning an Object-Oriented Programming Course”. The ACM Transactions on Computing Education 16 (4), 2016.

[6] M. Havenga, B. Breed E. Mentz, D. Govender, I. Govender, F. Dignum and V. Dignum, “Metacognitive and problem-solving skills to promote self-directed learning in computer programming: Teachers’ experiences.” Sa-educ Journal, 10(2), 2013, pp.1-14

[7] A. Robins, J. Rountree and N. Rountree, “Learning and teaching programming: A review and discussion.” Computer Science Education, 13(2), 2003, pp.137-172.

[8] A. Moreno, N. Myller, E. Sutinen, and M. Ben-Ari, “Visualizing programs with Jeliot 3.” In proceedings of the Working Conference on Advanced visual interfaces”, ACM, 2004, pp. 373-376.

[9] A. T. Virtanen, E. Lahtinen and H M. Järvinen, “VIP, a visual interpreter for learning introductory programming with C++” In proceedings of The Fifth Koli Calling Conference on Computer Science Education, 2005, pp. 125-130.

[10] S.H. Edwards and M.A. Perez-Quinones, “Web-CAT: automatically grading programming assignments.” In ACM SIGCSE Bulletin, ACM, Vol. 40, No. 3, 2008, pp. 328-328.

[11] M. Joy, N. Griffiths and R. Boyatt, “The BOSS online submission and assessment System”. ACM Journal on Educational Resources in Computing. 5(3), 2005.

[12] T. Boyle, “Design principles for authoring dynamic, reusable learning objects”. Australian Journal of Educational Technology, 19(1), 2003, pp. 46 – 58.

[13] A. Vihavainen, J. Airaksinen and C. Watson, “A systematic review of approaches for teaching introductory programming and their influence on success.” In proceedings of the Tenth Annual Conference on International Computing Education Research, ACM, 2014, pp. 19-26

[14] F.F.H. Nah, Q, Zeng, V.R. Telaprolu, A.P. Ayyappa and B. Eschenbrenner, “Gamification of education: a review of literature.” In International conference on hci in business (pp. 401-409). Springer, 2014.

[15] B.S. ChamAkpolat, and W. Slany, “Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification.” In Software Engineering Education and Training (CSEE&T), 2014 IEEE 27th Conference on, pp. 149-153.

[16] A. Knutas, J. Ikonen, U. Nikula, and J. Porras, “Increasing collaborative communications in a programming course with gamification: a case study”. In Proceedings of the 15th International Conference on Computer Systems and Technologies, 2014, pp. 370-377.

[17] M. Olsson, P. Mozelius and J. Collin, “Visualisation and Gamification of e-Learning and Programming Education.” *Electronic journal of e-learning*, *13*(6), 2015, pp.441-454.

[18] T. Rajala, M.-J. Laakso, E. Kaila and T. Salakoski, ”Effectiveness of Program Visualization: A Case Study with the ViLLE Tool.” *Journal of Information Technology Education*, *7, 2018*.

[19] D. Parsons and P. Haden, “Parson's programming puzzles: a fun and effective learning tool for first programming courses.” In Proceedings of the 8th Australasian Conference on Computing Education-Volume 52, 2006, pp. 157-163.

[20] E. Kaila, E. Kurvinen, E. Lokkila, M.-J. Laakso and T. Salakoski, ”Enhancing Student-Teacher Communication in Programming Courses: a Case Study Using Weekly Surveys”. In proceedings of ICEE 2015 - International Conference on Engineering Education

[21] M.-J. Laakso, E. Kaila, E. and T. Rajala, “ViLLE – collaborative education tool: Designing and utilizing an exercise-based learning environment.” Educ. Inf. Technol., 2018. https://doi.org/10.1007/s10639-017-9659-1

[22] E. Lokkila, E. Kurvinen, E. Kaila and M.-J. Laakso, “Automatic Recognition of Student Misconceptions in Primary School Mathematics.” In proceedings of EDULEARN15 - 7th International Conference on Education and New Learning Technologies, 2015.

[23] T. Rajala, E. Kaila E., M.-J. Laakso and T. Salakoski, “Effects of collaboration in program visualization.” Appeared in Technology Enhanced Learning Conference 2009 (TELearn 2009), October 6 to 8, 2009, Academia Sinica, Taipei, Taiwan

[24] T. Rajala, E. Kaila E., M.-J. Laakso and T. Salakoski, ”How does collaboration affect algorithm learning? A case study using TRAKLA2 algorithm visualization tool.” In proceedings of 2010 International Conference on Education Technology and Computer (ICETC 2010)

[25] T. Rajala, E. Kaila, J. Holvitie, R. Haavisto, M.-J. Laakso and T. Salakoski, “Comparing the collaborative and independent viewing of program visualizations.” In Frontiers in Education 2011 Conference, October 12-15, Rapid City, South Dakota, USA.

[26] T. Rajala, E. Lokkila, R. Lindén, M.-J. Laakso and T. Salakoski, ”Students’ perceptions on collaborative work in introductory programming course”. In ICEE 2015 - International Conference on Engineering Education.