Olympiads in Informatics, 2014, Vol. 8, 93–109

Utilizing an Exercise-Based Learning Tool Effectively in Computer Science Courses

Erkki KAILA, Teemu RAJALA, Mikko-Jussi LAAKSO,
Rolf LINDÉN, Einari KURVINEN, Tapio SALAKOSKI
University of Turku
e-mail: {ertaka, temira, milaak, rolind, emakur, sala}@utu.fi

Abstract. Educational technology and learning environments are becoming more and more common in all levels of education. Still, the main focus in research seems to be on which tools to use rather than how to effectively use them. In this paper, we first discuss the aspects that should be considered when adapting an exercise-based learning environment into curriculum. Based on our earlier research on the topic, we present three rules for adapting the tools. Next, a six-year study on using a learning environment in two courses is presented. Throughout the six course instances, the adaptation and integration of the tool is gradually altered. The results seem to confirm the positive effect of changes made in adaptation. When the three rules presented earlier are revisited in correlation with the results obtained, we can state that following the rules of adaptation lead to better student performance.

Keywords: learning environments, automatic assessment, course design, tool adaptation.

1. Introduction

According to multinational study by McCracken *et al.* (2001), programming is one the most difficult skills to acquire. There are various educational tools developed to aid the process, but comprehensive research about their pedagogical usage is still quite rare. Any tool or system, no matter how proficient, can only produce real educational value if adapted and utilized properly. In this article, we consider various factors that may have an effect on the efficiency of the tool usage: tool introduction, student engagement, motivation and reward. Based on our earlier research, most of these factors have a considerable effect on learning results. Hence, it seems that in addition to considering which tool to use, it's equally important to consider how to use it.

In this article, we present three rules for adapting a learning environment, based on our earlier experiments. Though named rules, they are actually ideas to consider when designing course structure and educational technology adaptation. We also present a comprehensive 6-year study, where a learning environment was used in two courses throughout six years. In latter instances, new exercise types were introduced to try to improve motivation. Some other changes in the tool and the usage were also introduced throughout the years. The holistic idea has been to gradually improve the tool adapta-

tion to find out how the learning effects and motivation can be maximized. The student performance and grades are presented to find out whether the changes had effect in particular years.

2. Literature Review

There are various learning environments developed over the years. Crescenzi and Nocentini (2007) present a two year experiment of adapting an algorithm visualization tool into a programming course. The student feedback was mainly positive, but they don't report any changes in student performance. Laakso *et al.* (2005) adapted an algorithm visualization tool called TRAKLA 2 into two courses at separate universities at Finland. They found out that the pass rate increased significantly, and the student feedback was mainly positive. Still, the same group (Laakso et al, 2009) found out later that using the same tool in collaboration with another student has an even higher positive effect on learning. Hence, anyone adapting a tool should be encouraged to find out further whether the positive effects can be enforced.

Educational technology can of course be used in all kinds of courses. De Lange *et al.* (2002) surveyed students' opinions on adaptation of WebCT on accounting course, and found out that their satisfaction with environment is tightly associated with lecture notes, forum, on-line assessment and other tools in that environment. Paechter *et al.* (2010) suggest that the key factor in affecting students' motivation is making the learning objects transparent and providing possibility for self-assessment. Self-assessment should hence have a major role in any exercise-based learning environment. Students' attitudes should have a considerable effect on adaptation: Saunders and Klemming (2003) reported a two-year experiment where they integrated technology into traditional learning environment, and found out that though the students found the module harder to complete than others, their performance was actually better. The cognitive load for adapting new tools (see for example Chandler and Sweller, 1996) is an issue that should be considered when designing technology enhanced curriculums.

Liaw *et al.* (2007) also surveyed the attitudes of students and educators towards e-learning, and found out that the instructors' attitudes are highly positive. The analysis on students' attitudes revealed, that an effective learning environment is influenced by learner autonomy and teacher help, among other things. Hence, it is important to remember that educational technology is not something that can be added into curriculum and then forgotten. Lockyer and Patterson (2008) in fact state, that "the lecturers may have to play a considerable technical support role in helping students who are new to such technologies".

There are other studies that emphasize the instructors' satisfaction in educational technology. For example, Zuvic-Butorac *et al.* (2010) present a huge effort of implementing an e-learning environment of more than 400 courses and 15,000 students in Croatia. The teachers' attitudes were surveyed and found out to be highly positive towards the technology. Still, as O'Neill *et al.* (2004) state in their literature review about eLearning implementation, if new technology is to be integrated into learning properly, comprehensive training and support for instructors should be provided.

3. How to Adapt an Exercise Based Learning Environment

3.1. *Selecting a Suitable Environment*

The first step in adaptation is selecting a proper environment. There are various issues that should be considered when selecting the tool. First – and probably the most important issue – is that the selected tool should provide adequate benefits for both the teacher and the student. As discussed earlier, the most obvious benefit for teacher is the time saved in assessing exercises and assignments. However, to gain any real benefit in time saving, the environment either needs to come with an existing set of usable exercises or the time cost of preparing the exercises needs to be tolerable. As stated in Naps *et al.* (2001), the lack of time is the most important reason for teachers not using visualization tools; the same can probably be applied to any learning environment.

From the student's point of view, the most obvious benefits come from automatic assessment and immediate feedback. The ability to do the exercises any place and any time, and still get supportive feedback, is something that is hardly possible with traditional methods. Improved learning results (Kaila *et al.*, 2009A) are also a significant benefit both for the student and the teacher. Evaluating the learning effects outside controlled studies might be difficult as there are various factors influencing the learning outcome. Still, as shown in Kaila *et al.* (2010), and furthermore in the later sections of this paper, it is possible to significantly improve the results in CS course if a learning environment is introduced and used properly.

There are also some technical issues that need to be considered when selecting an environment. First, there is the initial cost of tool utilization and management. Though most of the common learning environments can be adapted free-of-charge, there might be hidden costs, such as upgrading server equipment and training the users. If the tool is hosted externally, these costs can however be kept in minimum. Moreover, the technical requirements for using the tool should be evaluated beforehand. Some exercises may need plugins – such as Java or Flash – installed into browser before working properly. In some physical environments installing additional components may be difficult or impossible.

3.2. *Three Rules for Adaptation*

In this section, we present three rules that should be taken into account when adapting a learning environment into a course. The rules are based on our earlier results on the topic, and are revisited when the results of this research are discussed.

3.2.1. *Rule 1: Introduce and Integrate*

The first rule is that the tool should be properly introduced and integrated into course. We have previously studied the effects of cognitive load on students when using a visualization tool (Laakso *et al.*, 2008). In the study, the students who went through a comprehensive tutorial about using the tool statistically significantly outperformed the control group. Hence, we suggest, that a separate introductory session should be arranged before the tool is adapted into actual learning. The introduction should be made from two points

of view: technical and pedagogical. The technical introduction contains issues such as logging in and user interface. The pedagogical introduction should address issues such as the order and schedule of the exercises taken, using additional materials to assist learning, and the role of exercises as a part of comprehensive learning experience.

This leads as to the second part of this rule: we suggest that the learning environment should be properly integrated into the course. This means that the exercises in the environment should substitute and supplement the existing materials, where relevant. In practice, this may mean that the course needs to be partially redesigned. In Laakso *et al.* (2014) we presented a programming course reform, where half of the lectures were replaced with interactive tutorials that emphasized active and collaborative learning. The results were remarkable, as the dropout rate decreased and the grades improved statistically significantly after the change. Moreover, the students seemed to find the active approach more motivating and enjoyable.

3.2.2. *Rule 2: Engage the Students*

Naps *et al.* (2002) presented a hypothesis of engagement taxonomy, where they divided the usage of visualization tool into passive (no-viewing and viewing) and active (responding, changing, modifying and presenting). They suggested that using a visualization tool may only produce considerable learning if the tool is used in active levels. We later confirmed the hypothesis in Kaila *et al.* (2009B). We suggest that the results gained from using a visualization tool can be generalized to any types of exercises: if the students are engaged into active learning process, the results are better. Moreover, collaboration can be used to deepen the level of engagement. In Rajala *et al.* (2009) we found out, that if exercises are done in collaboration with another student, the learning results can be significantly improved. In Laakso *et al.* (2014) we presented a programming course reform (see previous Section), where collaboration was brought to classroom exercise sessions by introducing a collaborative mode in learning platform.

3.2.3. *Rule 3: Make it Mandatory, but Reward the Students*

As a third rule, we suggest that the usage of the tool should be made mandatory, but the students should still be rewarded from doing the exercises in the environment. A typical approach is to set minimum limits that need to be reached, and reward the students from exceeding that limit. The reward can be divided into two categories: an internal reward is something gained within the tool. Typically points are awarded when a student successfully completes an exercise or assignment. An external reward is something the students gain outside the learning environment. For example, the students may be awarded with grade improvement, bonus points for exam, or other forms of compensation from completing the exercises in the environment.

In Laakso *et al.* (2014) we present a case where students were required to complete at least five out of seven tutorials during the programming course. However, no minimum score limit was set. The students however completed a remarkable amount, 91% of all points on average, though reaching this amount meant doing extra work outside tutorial sessions. The students could pass the course without a final exam by completing at least 90% of all points awarded from all course components (including lectures, tutorials and course assignments). Still, of all students that reached the 90% level, only a handful skipped the final exam.

4. ViLLE

4.1. *Background*

ViLLE is a learning environment, developed at the University of Turku, Finland. It started out as a program visualization tool in 2004, and later expanded into comprehensive collaborative exercise and course management environment. From the beginning, ViLLE has been developed based on the research done. All major features have been tested with controlled experiments, and only the useful ones have been included in the published version. For example, the engagement taxonomy hypothesis (Naps *et al.*, 2002) lead into developing interactive questions into then-passive visualization tool, and the good experiences on collaborative use (Rajala *et al.*, 2009) encouraged us to develop collaborative mode that enabled two or more students working at the tasks together.

Since the earlier version was used in the first course presented in this paper, both versions are introduced separately.

4.2. *The Early Version of ViLLE – The Visual Learning Tool*

The first version of ViLLE is a program visualization tool (see Fig. 1) that can be used to display the execution of programs one row at a time. The execution is visualized with various components: the current and previous rows are highlighted, the variable states are displayed in their own area, and each subprogram with its local variables is displayed

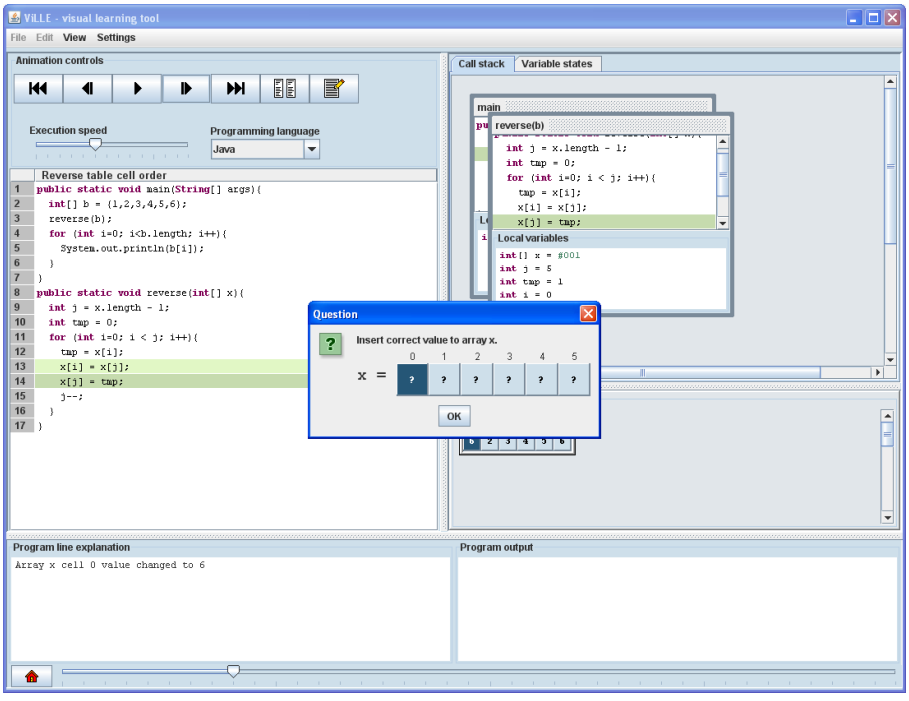


Fig. 1. ViLLE version 1: the student view displaying visualization exercise.

in a single frame in call stack and so on. Moreover, ViLLE displays a verbal explanation about the currently executed line. The tool supports a variety of imperative programming languages – including for example Java, Python and C++ – and automatically translates the programs written in Java to other supported languages. Students can view the execution in parallel view, which displays the executed program in two selectable languages at the same time.

To enhance active learning, the example programs can be accompanied with multiple choice questions or graphical array questions. The questions are inserted into desired steps in program. When a question is encountered the program execution halts until student gives an answer. ViLLE version 1 was deployed as a Java applet or Java application, but it could be connected to a TRAKLA II server (Malmi *et al.* 2004). In this case the server tracks student logins and all achieved points in different exercises. A complete description of the tool can be found in Rajala *et al.* (2007) and in Kaila *et al.* (2009A).

4.3. *ViLLE Now – a Collaborative Learning Environment*

As of 2009, ViLLE was expanded into an exercise-based collaborative learning environment. New client-server architecture was designed, with a focus on teachers' collaboration and with a support for various exercise types. In ViLLE version 2 (see Fig. 2), the teachers can use the built-in editors to create and edit virtual courses and assignments. Moreover, all content set as public can be browsed, utilized and modified by all other teachers registered in ViLLE. New exercise types for various topics were created. For programming, coding exercises, code sorting exercises and simulation exercises were designed among many others. Moreover, exercise types for mathematics, language

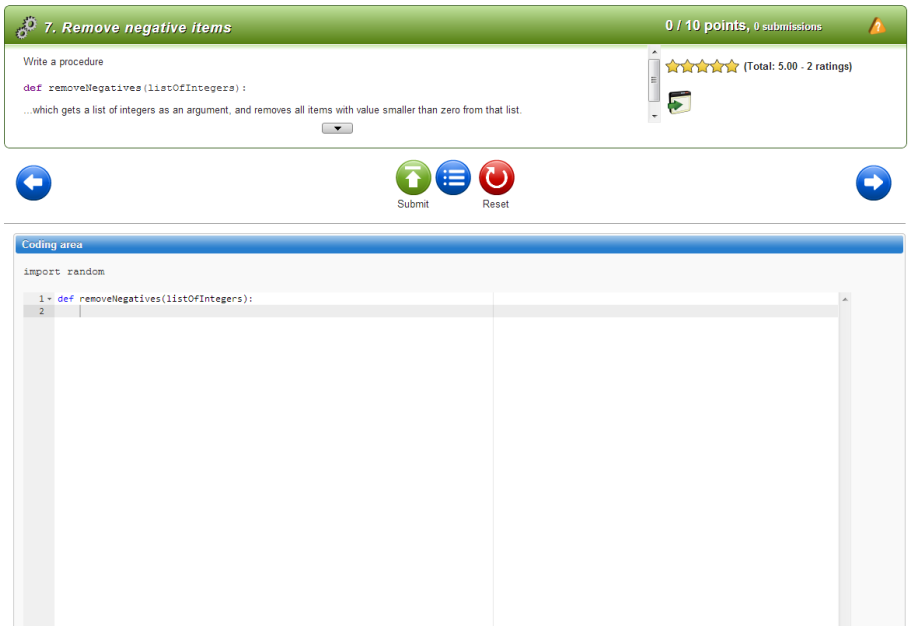


Fig. 2. Coding exercise in current version of ViLLE.

teaching and various other topics have been developed over the past few years. A comprehensive list about ViLLE exercise types can be found in Appendix A.

Since the new version introduced a dedicated ViLLE server, there was no more need to utilize the TRAKLA II server. ViLLE automatically collects a vast amount of data on student performance, including for example all achieved scores and time used to complete the exercises. Additional exercise specific data is also collected: for example, in visualization exercises ViLLE automatically saves all control usage data, including time stamps, when the student does the exercise. All data gathered can be viewed in ViLLE's statistical view by course's teachers.

The new version also supports collaborative learning where more than one student can join the same session. Besides exercises, there are various other tasks that can be used in courses: if accompanied with RFID readers, ViLLE can be used to easily record course attendances and demonstrations. It also supports study journals and course assignments, to name a few. All exercises, whether they are automatically assessed or not, can be used in electronic exams. It also has an editor for building tutorials that combine exercises with other materials, and a research project management system for research collaboration.

The complete description of the tool can be found in Laakso *et al.* (2014)**.**

5. Methodology

5.1. *Overview*

The research was carried between years 2007 and 2012. The data was collected from two separate courses: in the first course – observed in three instances between 2007 and 2009 – the version 1 of ViLLE was used, while in the second course – with three instances between 2010 and 2012 – the newer version 2 was utilized. The usage of tool varied in different instances of the course: the tool was adapted more thoroughly year by year. A gradual increase in the usage was justified by excellent results and feedback gathered from teachers and students.

5.2. *Course Instances*

The first course observed (from now on **Course 1**) was called an Introduction to Information Technology. The goal of the course is to teach computer science fundamentals as well as introductory programming concepts to CS majors at University of Turku. The course is somewhat typical introductory course in computer science, containing basic principles of algorithms and data structures, accompanied with programming fundamentals in Python. Three instances of the course were researched: in 2007, ViLLE was introduced to the course. The usage of the tool was not mandatory; instead, a link to exercises was provided in course web page. In two consecutive instances, 2008 and 2009, ViLLE was made a mandatory part of the course: if the students did not complete at least 40% of all ViLLE exercises, they failed the course. All course instances were taught by the same teacher, and no other significant chances between instances were made.

The second course observed (from now on **Course 2**) is called an Introduction to Programming. It is a mandatory course in Bioinformatics program at University of Turku, and aims at teaching basic programming concepts in Python. The course hence contains essential topics in imperative programming, such as variables, loops and functions, as well as some Python specific topics, but does not include object oriented programming. Three instances of Course 2 were also observed: at 2010 ViLLE was included – as mandatory component, but with visualization exercises only. In two latter instances various other exercise types were introduced as well. In the latest instance (2012) ViLLE was also used to keep track on lecture attendances and demonstration scores, with bonus awarded for good performance on these components. Moreover, in the last instance, ViLLE was also used as a platform for course final exam. As was the case with Course 1, all instances were taught by the same teacher, and no other substantial chances were made in course through these instances.

The usage of ViLLE throughout the course instances is displayed at Table 1**.**

5.3. *Exercises*

In Course 1, ViLLE was first introduced as an optional supplement. At two later instances the usage of the tool was made mandatory. A total of 60 exercises were divided into seven categories: variables and conditions, strings, loops, sub programs, arrays, recursion and sorting algorithms. Each exercise consisted of visualized program code and 5 to 10 questions. Each exercise was scored in scale of 0 to 10 based on the correctness of answers. All exercise rounds were open from the beginning of the course, and references to suitable exercises were made on other course materials.

In Course 2, ViLLE was mandatory in all three instances. At the first instance only visualization exercises were used – the exercise collection was roughly equivalent to the collection used in Course 1 with minor modifications. At the two latter instances other exercise types were introduced. The course was hence divided into eight exercise rounds, based on the topics in course: the first round was an introduction to ViLLE, and the latter rounds about variables and data types, strings, selection, loops, functions, lists and tuples, followed with a round of additional exercises. Each round consisted of five different types of exercises:

Table 1
Usage of ViLLE at course instances

Year	Course	ViLLE exercises	Mandatory
2007	Course 1	Visualization	No
2008	Course 1	Visualization	Yes
2009	Course 1	Visualization	Yes
2010	Course 2	Visualization	Yes
2011	Course 2	Various	Yes
2012	Course 2	Various, including other performance and course exam	Yes

- *Visualization exercises*: these were similar to exercises used in Course 1 and in the first instance of Course 2. A handful of existing visualization exercises were selected, of which some were slightly modified to suit the topics better.
- *Code sorting exercises*: exercises where code lines were shuffled into random order, and the student needed to sort them into correct order according to given task.
- *Puzzle exercises*: an exercise, where the student needs to combine for example variable types and value ranges or string operations and results.
- *Coding exercises*: an exercise where the student needs to write a program – or a missing part of the program – in Python to fulfil given task. The program written in ViLLE can be instantly translated and executed.
- *Quizzes*: ten multiple choice and open questions about the topic at hand.

In Course 2 the exercises were integrated into course curriculum more tightly. Each round of exercises was opened after the lecture about corresponding topic was given. The exercises were designed to cover all aspects of the topic at hand as thoroughly as possible. After opening, all rounds were open until the final exam.

5.4. *Method*

Since the research contains two different courses, only instances of the same course are compared. From each course instance, final grades were obtained. The experiment is a between-subject design with final exam results a dependent variable. ViLLE usage was the only significant between-subject factor (independent variable), since no other significant changes in courses during the observed period were made: the instances were taught by the same teacher, and there were no substantial changes in other course components or materials. Since Course 2 used the most recent version of ViLLE, we also had access to comprehensive exercise data on those instances; hence, statistics about ViLLE usage in Course 2 are also discussed.

6. Results

6.1. *Course 1*

All instances of Course 1 were graded on scale of one to five, five being the best. If the student did not pass the course, no grade was given. The final grade distribution in all course instances is displayed in Table 2 and visualized in Fig. 3**.**

As seen on Table 2, the pass rate and the average grade improved at latter instances, when the exercises were made mandatory. The grade distribution is visualized in Fig. 3.

As seen on Fig. 3, the amount of lesser grades (1, 2 and 3) is clearly smaller at the latter instances of the course, compared to first year when ViLLE exercises where optional. To confirm this, a chi test between course grade distributions was used to calculate the independence between all instances, using the formula

$$x^2 = \sum_{i=1}^{6}\sum_{j=1}^{2}\frac{(C1_{ij} - C2_{ij})^2}{C2_{ij}}$$

where $C1$ and $C2$ are the course instances compared. The results are displayed at Table 3.

As seen on Table 3, the grade distribution at first instance is independent, while latter instances seem to follow the same pattern more tightly.

Table 2
Grade distribution in Course 1 instances

	2007 (N=131)	2008 (N=134)	2009 (N=181)
5	34	40	46
4	17	19	27
3	9	20	33
2	21	16	25
1	25	15	23
Fail	25	24	27
Total passed	106	110	154
% of all passed	80.92 %	82.09 %	85.08 %
Grade mean	3.13	3.48	3.31

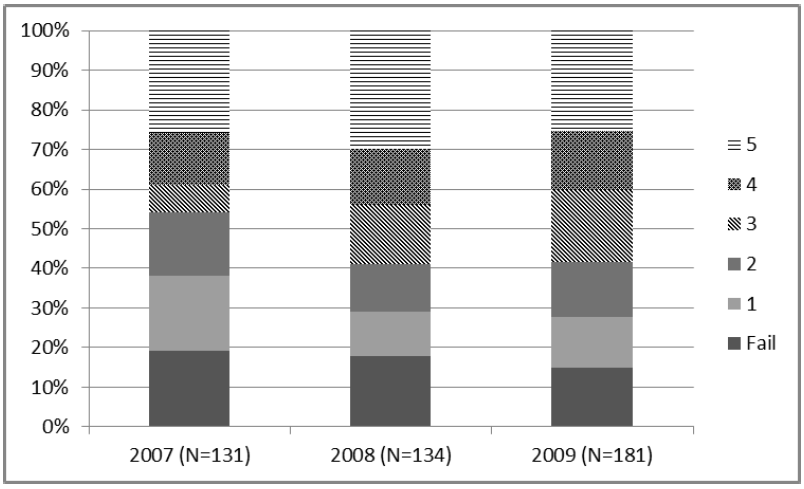


Fig. 3. Grade distribution at Course 1 visualized.

Table 3
Independence between grade distributions of Course 1 instances

Courses	2007 and 2008	2007 and 2009	2008 and 2009
x^2	0.002	<0.0001	0.022

6.2. *Course 2*

It is to be noted, that the number of students in all instances of Course 2 was rather small. Still, certain trends can be observed. Course 2 was also graded in standard scale of 1 to 5. The final grade distribution of all instances is displayed in Table 4.

The percent of students who passed the course has been extremely high in all instances. However, it seems that there is a trend to be seen on the average grades: the average is higher on the latter instances of the course, where varied types of ViLLE exercises were used. The grade distribution is visualized at Figure 4.

Since all of the instances did use ViLLE exercises, and in all instances the usage was required, course grade averages were also compared to earlier instances (<2010) of the course (see Table 5). However, since the teacher was different, and there were other minor changes in the course at 2010 as well, the data should be observed with caution.

Points gathered from ViLLE exercises in all instances of Course 2 are displayed at Table 6.

Table 4

Grade distribution in instances of Course 2

	2010 (N=23)	2011 (N=16)	2012 (N=25)
5	10	10	16
4	3	1	2
3	4	1	3
2	2	1	1
1	3	1	2
Fail	1	2	1
% of all passed	95.65 %	87.50 %	96 %
Grade mean	3.52	3.75	4.04

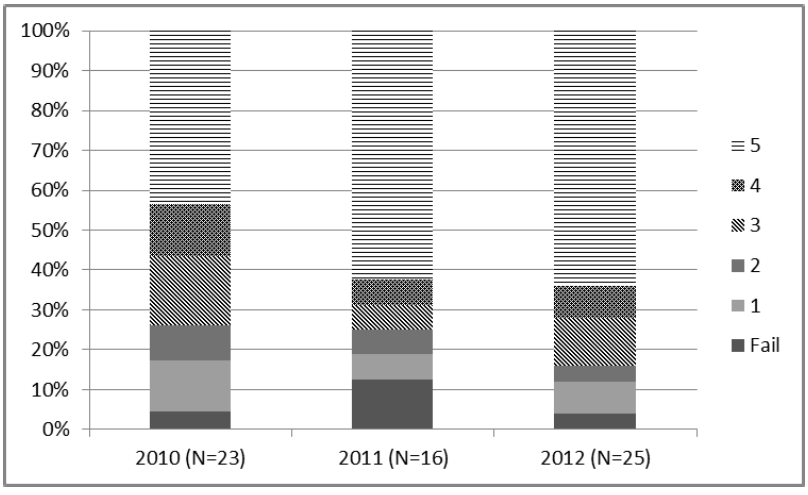


Fig. 4. Grade distribution at Course 2 visualized.

Table 5
Course 2 instances' mean grades throughout 2006…2012

Year	Grade mean
2006…2009 (N=21)	3.14
2010 (N=23)	3.52
2011 (N=16)	3.75
2012 (N=25)	4.04

Table 6
Points gathered in ViLLE in instances of Course 2

Year	Total maximum	Mean score	Std. dev.	% of maximum
2010	700	552.36	99.57	78.91 %
2011	660	567.06	128.06	85.92 %
2012	660	588.73	83.39	89.20 %

Though the differences are rather small, it seems that the students completed more exercises when visualization was accompanied with other exercise types starting from 2011.

7. Discussion

Based on the results presented, it seems that ViLLE exercises have a positive effect on learning. The average grade in both courses increased – though in Course 2 there are no significant changes (though this might be because of the low N). The pass rate in Course 1 also improved. In this section, the results for both courses are first discussed separately. Then the rules for adaption presented earlier are revisited in context of the results. Finally, as there are issues when measuring and comparing the performance at whole course level, some critical points of view are presented.

7.1. *Performance at Course 1*

Three instances of Course 1 were observed: at the first instance (2007) a link to ViLLE applet was given to students at course web page, but no points were collected and hence no minimum score limits set. In consecutive instances (2008 and 2009) ViLLE was made mandatory at course, as the minimum of 40 % of all points in ViLLE needed to be collected to pass the course. Based on the results, it seems that this had an effect on learning results. The mean average increased, and the amount of lower grades (1, 2 and 3) decreased. Also, the passing percent increased from 80.92 % to 82.09 % and 85.08 %, respectively.

It seems, that the visualization exercises combined with active learning in the form of questions has a positive effect on results. We have previously shown (see e.g. Kaila *et al.* 2009A, Laakso 2010), that visualization seems to have a highly positive effect on learning. It seems that the results gained from controlled two hour experiments can be generalized to learning at whole course. This also seems to confirm our earlier results on high school level programming course (Kaila *et al.*, 2010).

7.2. *Performance at Course 2*

There were also three instances observed in Course 2. In all of them ViLLE was made a mandatory part of the course, with minimum amount of 50 % of all points to be gathered to pass the course. The difference between instances was that at two latter instances (2011 and 2012) new exercise types were presented: only a handful of earlier visualization exercises were kept and four new exercise types were presented.

The same trend seems to exist at Course 2 results as well: when compared to earlier instances with no ViLLE (2009 and earlier) of the course, the grade mean seems to be higher when ViLLE exercises were used. Moreover, it seems that at the latter instances (2011 and 2012) of observed courses the distribution of grades seemed to focus more on the higher level of grades. No statistical differences could be found between groups, though one possible reason for this might be the low N. The trend in number of exercises completed at latter instances is still interesting: the students seemed to do more of the exercises when new types were introduced among the visualization. It is likely, that more heterogeneous set makes doing the exercises more motivating.

7.3. *The Rules of Adaptation Revisited*

The **first rule** we presented about adapting learning technology was to *introduce and integrate.* The results from Course 1 seem to underline this: when ViLLE was presented as an external tool with no connection to course otherwise, it did not seem to have a strong effect on learning. When the tool was made a mandatory part of the course, with connections drawn to other material, the grade and pass rate got higher. In Course 2 the introduction and integration was even tighter: there was a special introductory round in ViLLE where the exercise types were presented. Moreover, the exercise rounds in ViLLE were tightly integrated into course curriculum. Each round was opened after the lecture about the topic was given.

The **second rule** was to engage the students. The engagement taxonomy presented by Naps *et al.* (2002) states, that higher the level of engagement, the better the learning results. In latter instances of Course 2, new exercise types were presented. While visualization exercises lie in the engagement level of responding, most of the new types are on the higher levels of engagement. Based on the results, it seems that the students were more motivated in doing the exercises after the change, and it also seems, that the learning results were better. Though, as mentioned before, no statistically significant differences were found due to low number of students in course.

The **final rule** was to make the tool mandatory, but reward the students on using it. This rule was adapted on two final instances of Course 1 and in all instances of Course 2. In Course 1 the effect can be clearly seen: the results got better as soon as the tool was made mandatory. It is possible, that not all students find the visualization exercises motivating enough to complete them on their own. It is also likely, that at least the weaker students might not have enough patience to go through the more difficult exercises if they are not required. In Course 2 bonus points for final exam were rewarded if enough ViLLE points were gathered. This also seemed to have a motivating effect, as seen on scores obtained in ViLLE exercises: the 50 % minimum limit was clearly exceeded in all instances of the course.

7.4. *Issues in Course Long Performance Measurement*

There are some known issues when measuring the learning effects throughout the course. First, there are usually several factors that affect the learning results. In both courses, other variables were kept as steady as possible: the same teacher taught all instances of both courses and no significant changes in materials or course curriculum were made between instances. Still, isolating all factors that affect the learning is practically impossible. Also, measuring the actual learning outcome is difficult. The best we can do on course level is to compare the total grades obtained from course. As long as the components affecting the grade – and the components used to measure the grade – are kept somewhat similar, the mean grade should be reliable enough, – especially if the number of students in the course is high enough.

References

Chandler, P., Sweller, J. (1996). Cognitive load while learning to use a computer program. *Applied Cognitive Psychology*, 10, 151–170.

Crescenzi, P. and Nocentini, C. (2007). Fully integrating algorithm visualization into a cs2 course: a two-year experience. In: *Proc. of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* : *Dundee, Scotland, June 25–27, 2007*. ITiCSE '07, ACM, New York, NY, 296–300.

DeLange, P., Suwardy, T., Mavondo, F. (2001). Integrating a virtual learning environment into an introductory accounting course: determinants of student motivation. *Accounting Education*, 12(1), 1–14.

Kaila, E., Rajala, T., Laakso, M.-J., Salakoski, T. (2009A). Effects, experiences and feedback from studies of a program visualization tool. *Informatics in Education*, 8(1), 17–34.

Kaila, E., Laakso, M.-J., Rajala, T., Salakoski, T. (2009B). Evaluation of learner engagement in program visualization. In: *12th IASTED International Conference on Computers and Advanced Technology in Education* (*CATE 2009*) : *November 22–24, 2009, St. Thomas, US Virgin Islands*.

Kaila, E., Rajala, T., Laakso, M.-J., Salakoski, T. (2010*)*. Long-term effects of program visualization. In: *12th Australasian Computing Education Conference (ACE 2010)* : *January 18–22, 2010, Brisbane, Australia*.

Laakso, M.-J., Salakoski, T., Grandell, L., Qiu, X., Korhonen, A., Malmi, L. (2005). Multi-perspective study of novice learners adopting the visual algorithm simulation exercise system TRAKLA2. *Informatics in Education*, 4(1), 49–68.

Laakso, M.-J., Rajala, T., Kaila, E. and Salakoski, T. (2008). The impact of prior experience in using a visualization tool on learning to program. In: *Proceedings of CELDA 2008*. Freiburg, Germany, 129–136

Laakso, M.-J., Salakoski, T., Grandell, L., Qiu, X., Korhonen, A. and Malmi, L. (2005*)*. Multi-perspective study of novice learners adopting the visual algorithm simulation exercise system TRAKLA2. *Informatics in Education*, 4(1), 49–68.

Laakso, M.-J. (2010). *Promoting Programming Learning. Engagement, Automatic Assessment with Immediate Feedback in Visualizations*. TUCS Dissertations no 131.

Laakso M.-J., Kaila E. and Rajala T. (2014). Ville – collaborative learning environment. Sent to *Computers and Education*.

Liaw, S.-S., Huang, H..M. and Chen, G.-D. (2007). Surveying instructor and learner attitudes toward e-learning environments. *Computers & Education*, 49(4), 1066–1080.

Lockyer, L., Patterson, J. (2008). Integrating social networking technologies in education: a case study of a formal learning environment. In: *Proceedings of 8th IEEE international conference on advanced learning technologies.* Santander, Spain (2008), 529–533.

Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppälä, O., Silvasti, P. (2004). Visual algorithm simulation exercise system with automatic assessment : TRAKLA2. Informatics in Education, 3(2), 267–288

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., Velázquez-Iturbide, J. Á. (2002). Exploring the role of visualization and engagement in computer science education. In: *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, 35(2), 131–152.

O'Neill, K., Singh, G., O'Donoghue, J. (2004). Implementing e-learning programmes for higher education : a review of the literature. *Journal of Information Technology Education*, 3, 312–23.

Paechter, M., Maier, B, Macher, D. (2010). Students expectations of, and experiences in e-learning : their relation to learning achievements and course satisfaction. *Computers & Education*, 54, 222–229.

Rajala, T., Kaila, E., Laakso, M.-J., Salakoski, T. (2009). Effects of collaboration in program visualization. In: *Technology Enhanced Learning Conference 2009* : *TELearn 2009, October 6 to 8, 2009, Academia Sinica, Taipei, Taiwan*.

Rajala, T., Laakso, M.-J., Kaila, E., Salakoski, T. (2007). VILLE – a language-independent program visualization tool. In: Lister, R. and Simon (Eds.) *Koli Calling 2007 – Proceedings of the Seventh Baltic Sea Conference on Computing Education Research*, *Koli National Park, Finland, November 15–18, 2007*. (Conferences in Research and Practice in Information Technology, 88). Koli National Park, Finland, ACS.

Saunders, G., Kelmming, F. (2003). Integrating technology into a traditional learning environment. *Active Learning in Higher Education*, 4, 74–86.

Zuvic-Butorac, M., Nebic, Z., Nemcanin, D. (2011). Establishing an institutional framework for an e-iearning implementation –experiences from the University of Rijeka, Croatia. *Journal of Information Technology Education*, 10, 44–56.

Appendix A. ViLLE Exercise Types

Exercises for Computer Science

Visualization exercise: Combines the graphical, step-by-step execution of the example program with three types of questions: multiple choice questions, open questions and array questions.

Code sorting exercise: Commonly known as Parson's puzzle: the students need to arrange the shuffled program code lines into correct order so that given task is fulfiled.

Coding exercise: The task is to write a program – or a missing part of the program according to given specifications. ViLLE supports a variety of programming languages, including Java, C++, C# and Python.

Robot exercise: The goal of the exercise is to move number of boxes into specified target locations. The boxes are moved by writing an algorithm that controls a robot crane. Idea is to teach loops and methods in Java.

Clouds & Boxes: Reverse-visualization type exercise: the students are supposed to simulate the state of program after each step executed.

Other CS exercises: In addition, there are exercise types for testing binary calculations and conversions between hexadecimal, decimal and binary.

Exercises for Mathematics

Math exercises for elementary school level: There are several exercise types meant for teaching elementary level mathematics. In these exercises, the students for example need to find out the missing number, drag and drop numbers into number line, do long division, find out values in bar charts, calculate with fractions and so on.

Math exercises for higher levels: There are also exercise types meant for students in higher levels: for example, solving quadratic and first degree equations, doing differential coefficient calculations and writing inequality equations and sign charts.

Other Exercises

Quiz: The most basic exercise type: contains multiple choice and open questions with attachable materials. Quizzes can be utilized in any level and topic.

Survey: Can be used for course opening and closing surveys. Moreover, ViLLE surveys are typically utilized to implement assignments that are graded by teacher, for example essays.

Sorting: ViLLE contains exercise types for image puzzles, and for general sorting and pair matching of textual items.

Language exercises: There are several exercise types meant specifically for language teaching (such as fill-in, dialog, word ordering, punctuation and case, vocabulary test, compound exercise and so on). However, most of these can be utilized under other topics as well.

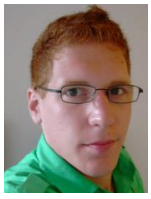
Image tagging: Exercise where the students need to identify areas in given or uploaded image.

E. Kaila, M.Sc., is working on his PhD about utilizing learning environments and assignments effectively. He has 25 scientific publications in peer-reviewed journals and conference proceedings. His research interests include program visualization, learning environments, automatic assessment and course design utilizing new technologies. He has been working on development of ViLLE from the beginning of the project.

T. Rajala, M. Sci., is a university teacher in Software Engineering at University of Turku, Finland. In addition to teaching, his work and research mainly focuses on developing educational software tools and studying their effectiveness in learning programming and algorithmic problem solving. Rajala finished his master's degree at University of Turku in 2007. He has 25 scientific publications in peer-reviewed journals and conference proceedings. He has also been working on ViLLE project since the beginning.

M.-J. Laakso, PhD(tech), is a lecturer and adjunct professor at Department of Information Technology at University of Turku. He has more than 35 scientific publications in internationals journals and conference proceedings. His main research interests are educational technologies, learning environments, automated assessment, visualization, immediate feedback, eAssessment and effect of collaboration in aforementioned topic. He is heading ViLLE team research group at University of Turku studying all these aspects and developing ViLLE – the collaborative education platform.

R. Lindén, M.Sc., is working on his PhD about automated student profiling and counselling. His research interests involve data mining, systems analysis and graph theory. He has three publications in peer-reviewed journals and conference proceedings. Lindén has been working on ViLLE project since the beginning of 2012.



E. Kurvinen, M.A. (Education), is finishing his M.Sc. in computer science, and starting his PhD about recognizing learning disabilities in mathematics. His research interests mainly concern usage of educational technology in mathematics education in all levels. Kurvinen has been working on ViLLE project since the beginning of 2012.

T. Salakoski, PhD, is a professor of Computer Science at University of Turku. He is the Dean of Science and Technology Education at the university, and the Head of the Department of Information Technology. He has more than 200 scientific publications in international journals and conference proceedings, and has supervised more than 10 PhDs and numerous MScs. He serves in scientific editorial boards and has organized and chaired international conferences. He is heading a large research group studying machine intelligence methods and interdisciplinary applications, especially information retrieval and natural language processing in the biomedical and health care domain as well as technologies related to human learning, language, and speech.