Pedagogical Experiences of Competitive Elements in an Algorithms Course

Fredrik Heintz and Tommy Färnqvist, *Linköping University, Sweden*

***Abstract*—We claim that competitive elements can improve the quality of programming and algorithms courses. To test this, we used our experience from organising national and international programming competitions to design and evaluate two different contests in an introductory algorithms course. The first contest turned lab assignments into a competition, where two groups ran competitions and two were control groups and did not compete. The second, voluntary, contest, consisting of 15 international programming competition style problems, was designed to support student skill acquisition by providing them with opportunities for deliberate practise. We found that competitive elements do influence student behaviour and our main conclusions from the experiment are that students really like competitions, that the competition design is very important for the resulting behaviour of the students, and that active students perform better on exams.**

We also report on an extra-curricular activity in the form of a semester long programming competition as a way of supporting student's deliberate practise in computer programming.

***Index Terms*—Computer science education, Programming competitions, Skill acquisition, Deliberate practise**

I. Introduction

Writing computer programs is a craft. Academic courses in computer programming teach basic concepts and fundamentals, but to become a good programmer a great deal of time dedicated to training with an emphasis on quality has to be invested by the student. There are studies indicating that it takes around ten years to transform a programming novice into an expert [8]. This view is supported by Ericsson et al., according to whom, approximately 10000 hours of *deliberate practise* is required to become an expert in an area [3]. Our experience from organising programming competitions at local, national, and international levels is that such competitions stimulate and inspire students to solve programming problems by themselves, thereby practising both problem solving and programming. This increases the students' skills as programmers and problem solvers, which substantially increases their employability – a fact which is evidenced at the international level, where top students are offered trainee positions at market leading companies, and at the local level, where we have been contacted numerous times about students participating in our competitions.

This work was supported in part by the Pedagogical Development Group at The Institute of Technology, Linköping University.

Fredrik Heintz, fredrik.heintz@liu.se, Department of Computer and Information Science, Linköping University, Sweden.

Tommy Färnqvist, tommy.farnqvist@liu.se, Department of Computer and Information Science.

We have investigated two different ways of using competitive elements for supporting student's deliberate practise in computer programming: Extra-curricular activity in the form of a semester long programming competition and as parts of a Data Structures and Algorithms course.

A. *Deliberate Practise*

To become an expert it is necessary to engage in deliberate practise, activities that are designed to lead to improvements of specific aspects of performance [3]. These activities should stretch an individual just beyond his or her current abilities, provide immediate feedback, be repeated multiple times, and require significant effort and full concentration.

A theory which supports the creation of appropriate deliberate practises is the skill acquisition model of Dreyfus and Dreyfus [1]. According to it, a student normally passes through five developmental stages, designated novice, competence, proficiency, expertise and mastery. In early stages detailed instructions are required while in later stages a tacit understanding of how to use the skill to achieve desired results even in novel situations has been developed. This implies that the type of activities and feedback changes when progressing through the stages.

II. Case Studies

There have been a considerable number of attempts in higher education institutions to build applications for automated assessment for different types of assignments. Due to the specific nature of programming assignments, automated evaluation of user submitted programs is fairly natural and has been used for over forty years. In particular, such use not only saves valuable instructor time, but also ensures impartial and immediate feedback on programs submitted. For example, Enström et al. [2] describe using automated assessment of lab assignments. Guerreiro and Georgouli [5, 6] additionally used automated judging for self-assessment purposes and Gárcia-Mateos and Fernández-Alemán [4] tried out replacing the final exam in a course with a series of activities involving a web-based judge.

A. *Local Championship in Algorithmic Problem Solving*

To provide appropriate deliberate practise to all students we have started a department championship in algorithmic problem solving. To this end, we have created a web-based system, interfacing the UVa Online Judge [7]. The current competition is individual and runs the whole semester. Every week each student receives three selected problems of varying difficulty with a common theme. A student can also at any time create a *challenge* of a particular difficulty, which means that a problem of the desired difficulty is randomly selected from a large database. To support feedback directly from other students, the contest design encourages other students to solve the same problems and discuss these in a forum. To support detailed feedback on solutions, threads in the forum can be locked to be viewable by only those that have solved the problem being discussed. Students therefore both get immediate feedback from an automatic judge and detailed feedback from staff and other students. Multiple related problems of increasing difficulty provide repetition and stretch the students' abilities. Since solving these problems require intense effort and concentration all the conditions for deliberate practise are satisfied by the competition.

Semester	Students active/registered	Problems solved by all/by winner	Solved problems per active/registered student	Unique problems solved
Spring 2010	12/12	81/15	6.75/6.75	16
Fall 2010	16/38	224/42	14/5.89	42
Fall 2011	27/82	356/84	13.19/4.34	215
Spring 2012	23/57	405/89	16.61/7.11	133

Fig. 1. Overview of activity in department championship.

The design of the competition influences the student activity, as Fig. 1 shows. When it only had 3 weekly problems, the maximum number of problems solved was relatively limited (spring and fall 2010). When challenges replaced weekly problems (fall 2011) the number of problems increased significantly, but the opportunities for discussion were reduced as few students solved the same problem. When challenges and weekly problems were combined (spring 2012) the competition could get the best of both designs. Informal feedback from the students also reinforce our perception that the competition has evolved to become better balanced, both in terms of difficulty and required effort as well as being able to keep the interest up during a longer period of time.

B. *Competitive Elements in a Data Structures and Algorithms Course (DALG)*

The DALG course is given at the start of the second year for students from the three main Computer Science curricula at our university, and is organised in traditional monolithic form with weekly lectures, class room sized tutorials, laboratory sessions, and a written final exam. The course comprises 6 ECTS credits, and during fall 2011 there were 140 students enrolled. There is one failing grade "U", and passing grades are designated 3, 4, and 5.

TABLE I Average credits taken and average grades in programming, math, and CS courses for different groupings

	All	Group 1	Group 2	Group 3
Total credits	42.4	43.6	43.4	40.0
Prog. credits	11.0	10.9	11.1	11.0
Prog. grade	3.73	3.75	3.84	3.58
Math credits	15.0	16.2	16.1	12.4
Math grade	3.61	3.57	3.65	3.60
CS credits	15.1	14.6	14.8	16.0
CS grade	3.78	3.75	3.84	3.74

1) *Lab Assignment Contests*

To compare different designs for the lab assignment contest we had four different groups. The first group competed based on speed (days from start of the course) and correctness (+3 penalty days for incorrect submissions). The second group competed based on quality (cyclomatic complexity and instruction count) and efficiency (runtime and memory consumption). The third and fourth groups were control groups and did not compete. Table I gives various background statistics about the groups at the start of the course. We have performed extensive statistical testing and found that the only significant difference (at the 5% level) between any groupings in Table I concerns math credits taken for Group 1 and Group 3, where both the means as well as the entire populations are significantly different. Group 4 consisted of students from our IT programme, while the other groups contained a mixture of students from our two other main CS programmes. The IT students take a rather different set of courses compared to our other CS students, and also incorporate problem based learning in an essential way, making it nonsensical to include their results from the first year in the comparison.

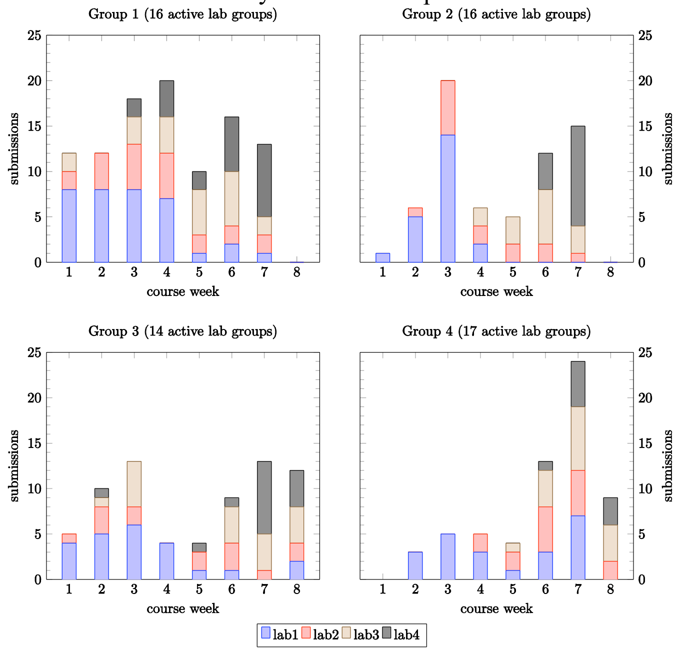

Fig. 2. Submission activity for the four groups on the four lab assignments in the DALG course.

All groups had to submit their labs to an online system as soon as they thought they were ready. We found that both competing groups were influenced by our competitions, with the strongest effect being that the submission pattern for the groups differs dramatically. This is shown in Fig. 2, where group 1 worked very fast and group 2 slightly faster than the control groups (group 3 and group 4).

In particular, the lectures did not go through the material for the first lab assignment until week two of the course, so the submission pattern of group 4 is more like what you would expect during a year with no competitions. We believe that since group 3 consisted of students from the same programmes as the competing groups the thrill of competition more easily spilled over to group 3 than to group 4. There were also large variations in the quality of the code between the groups.

2) Voluntary Contest

We have used our system, mentioned in II.A, to provide a voluntary contest in the DALG course. After each lecture the students enrolled in the contest received a challenge with a programming problem to solve. The problems were selected to either reinforce and repeat a topic just addressed in the lecture or to stretch the students' abilities by demanding use of deeper knowledge. The online judge used provides immediate feedback and since the contest requires students to solve challenges as quickly as possible due to time penalties otherwise incurred, it should prove to be an intensive experience. All together this contest also satisfies the conditions for deliberate practise.

TABLE II Average passing grade on final exam for different groupings.

Student group	Total number	Took exam	Passed exam	Average grade
All	140	118	95	3.36
Answered questionnaire	79	74	62	3.47
Completed all labs	76	73	62	3.47
Registered for voluntary contest	30	29	27	3.56
Solved at least one task in contest	15	15	15	3.6

Table II shows that 30 students registered for the voluntary contest and that 15 solved at least one task. The tendency that the average grade rises with rising commitment to our extra activities is not strong enough to be significant if we only look at the averages. However, the distribution of grades for students either enrolled in the voluntary contest and/or solving at least one task in the contest is significantly different (at the 10% level) from the grade distribution of all students.

TABLE III Average credits taken and average grades in programming, math, and CS courses for different groupings

	All	Answered questionnaire	Completed all labs	Reg. for contest	Solved at least one task
Total credits	42.4	45.7	46.2	45.4	50.0
Prog. credits	11.0	11.4	12.1	12.6	13.5
Prog. grade	3.73	4.00	3.83	4.00	4.27
Math credits	15.0	16.5	16.1	15.8	16.9
Math grade	3.61	3.68	3.70	3.68	3.89
CS credits	15.1	15.6	16.4	16.9	18.4
CS grade	3.78	4.06	3.89	4.04	4.26

Inspecting Table III, it would seem that an explanation for students enrolled in the voluntary contest performing better on the final exam could be that they are already stronger students, based on their past performance. Indeed, the average number of total credits taken by students solving at least one task is significantly larger (at the 5% level) than the average number of total credits over all students. The same holds for credits from programming courses for students registered for the contest and/or solving at least one task when compared to all students, as well as for CS credits for students solving at least one task. The average programming grade for students solving at least one task is significantly different from the average programming grade of all students (at the 10% level). No other such explanatory significant differences were found. It would seem that, statistically, the whole effect of students performing better on the final exam if they were enrolled in the voluntary contest cannot be explained by the stronger background of these students.

3) Questionnaire

At the end of the course we asked the students to fill out a questionnaire containing different questions depending on their involvement in different course activities. 79 out of 140 students responded. With respect to the background variables in Table III, the only difference between students answering the questionnaire and all students was that they have significantly different average CS grades (at the 10% level).

We posed both multiple choice questions as well as questions allowing free text answers. Due to space constraints we are only able to relate a few of the more important ones here. On the question "What is your general attitude towards the DALG contests", 22% answered very positive, 35% fairly positive, 30% neutral, 11% fairly negative, and 0% very negative, reinforcing the feeling of the course assistants and the authors that our experiments were received well in general. It might seem strange that so many students were positive towards competitive elements in courses while relatively few of them actually participated actively in the activities proffered. The explanation for this can be found in the answers to the free text questions, where students cite lack of time and /or lack of incentive in form of credits or points on the exam as the main reason for not participating actively. We also want to mention that general student satisfaction with the course, as evaluated by university central instruments, remained at the same (high) level as previous years.

III. Conclusion

Our main conclusions are that students enjoy competitive elements in programming courses and that the competition design highly influences the student behavior. We will continue studying how to design competitions to achieve particular goals such as increasing the number of students that pass the final exam in the DALG course.

References

[1] Dreyfus, S. E. and Dreyfus, H. L, "A five-stage model of the mental activities involved in directed skill acquisition". Technical report, 1980.
[2] Ensröm, E., Kreitz, G., Niemelä, F., Söderman, P., and Kann, V., "Five years with Kattis – using an automated assessment system in teaching". In *Proc. of the IEEE Frontiers in Education Conference (FIE)*, 2011.
[3] Ericsson, A., Nandagopal, K., and Roring, R., "Toward a science of exceptional achievement: Attaining superior performance through deliberate practice". *Annals of New York Academy of Science*, 2009.
[4] Gárcia-Mateos, G. and Fernández-Alemán, J., "A course on algorithms and data structures using on-line judging". In *Proc. of the 14th annual ACM SIGCSE conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pp. 45–49, 2009.
[5] Guerreiro, P. and Georgouli, K., "Combating anonymousness in populous CS1 and CS2 courses". In *Proc. of the SIGCSE conference on Innovation and Technology in Computer Science Education (ITiCSE)*, 2006.
[6] Guerreiro, P. and Georgouli, K., "Enhancing elementary programming courses using E-learning with a competitive attitude". *International Journal of Internet Education*, pp. 38–46, 2008.
[7] UVa Online Judge, 2012. http://uva.onlinejudge.org/
[8] Winslow, L. E., Programming pedagogy - a psychological overview. *ACM SIGSCE Bulletin*, vol. 28, no. 3, pp. 17–22, 1996.