Evolving an Introductory Programming Course: Impacts of Student Self–Empowerment, Guided Hands–On Times, and Self–Directed Training

Patrick Seeling
Department of Computer Science
Central Michigan University
Mount Pleasant, MI 48859
pseeling@ieee.org

***Abstract*—In this paper, the impacts that the evolutionary development of an introductory programming course has on student achievements is described. For the introduction of pass/fail grading with student self–empowerment through tokens, a significant increase in performance for programming assignments, quizzes, and total semester scores can be found. Furthermore, increased times of instructor–guided hands–on programming exercises have a positive effect as well. Thirdly, a strong correlation between how well students perform on an automatically scored self–directed online programming training environment and their achievements in programming assignments, quizzes, and the entire semester can be observed. In turn, the inclusion of these types of environments can benefit students significantly while introducing only little overhead for course instructors. The active utilization of these additional hands–on training options represents a significant potential to increase the total semester performance and should be actively communicated to students.**

***Index Terms*—Computer science education; Programming; Learner performance; Grading; Tokens**

I. Introduction

The fields of computer science, computer engineering, electrical engineering, information systems, and information technology commonly require foundational programming knowledge and several introductory programming courses are typically offered for learners in these domains. Introductory programming courses oftentimes are additionally offered to the broader student populations as well, motivated by the increased computational thinking demands and software–centrality found across disciplines. This trend is typically taken into account through making an introductory programming course part of the general education mix (or even a requirement). However, a significant rate of failure (e.g., about 25 % as in [1]) in these courses has received significant attention in the research community. The broad diversity of students enrolled, for example, could be regarded as one contributing factor to these high course failure rates [2]. Different strategies have been developed over the years to combat the problems associated with such a first introduction to programming. The result is a broad and growing body of research, which can only be regarded in a non–exhaustive fashion here due to space constraints. A criterion–referenced scheme was proposed in [3], which employs assessment strategies to overcome problems associated with different levels of student achievements. While showing promising results, individualized assessment might be out of scope for larger student bodies in traditional course offerings. Supporting a more hands–on flipped classroom approach can yield significant improvements in this respect, see, e.g., [4]. To enable additional hands–on time and repeated training without increasing the burden of instructors through additional grading of student program submissions, automatic grading systems have emerged, see, e.g., [5]. Students utilizing the feedback system for assignments in [5] made more submissions and leveraged the feedback provided for improvement. These commonly web–based systems allow for the repeated refinement of student submissions in feedback loops as well as fine–tuning the system usage [6].

An additional consideration in the context of introductory courses is to evaluate issues of student motivation. External rewards can have significantly detrimental effects on the students' self–reported interests when compared to intrinsic rewards [7]. In [8], tokens are successfully employed to increase the levels of intrinsic motivation. If misaligned, however, these tokens can be detrimental as well. Novices to performing programming assignments can significantly underestimate the time required for successful completion. The self–set goals and assignment deadlines might be poorly set by learners, commonly first–year college students [9]. The combination of instructor–set deadlines with an added flexibility of extension through tokens (to alleviate some of the potential pitfalls associated with a lack of self–scheduling skills) is a component of the course development described here. Due to space constraints here, an in–depth review of the student attitudes is out of the scope of this paper and the focus is on the overall impacts on academic achievement.

The remainder of this paper is structured as follows. In the next section, the overall course and modifications performed to evolutionarily upgrade the instruction and grading approaches over time are described. Next, a description of major findings from comparing student achievements is provided in Section III before the presentation concludes in Section IV.

TABLE I
COURSE FEATURES BETWEEN THE TWO OFFERINGS.

Category	Fall 2014	Spring 2015
Sections	3 short, 1 long hands-on	
Hands-on instruction	TA1 (short), Instructor (long)	TA2 (all)
Tic-Tac-Toe	Middle	End
Grading	Relative	Pass/Fail
Tokens	No	Yes

II. COURSE OVERVIEW

The Department of Computer Science at Central Michigan University is home to a traditional Computer Science and an Information Technology program. Students from both degree paths as well as Engineering students are required to take this initial programming course. In addition to this domain–specific student population, students across campus can enroll into the course to satisfy parts of their general education requirements. In turn, the prior knowledge levels and interests for further computer science studies of students enrolled in the course varies highly. The overall course features and differences between the two evaluated offerings are summarized in Table I and briefly described in the following.

A. Course Details

The course employs the JAVA programming language with an imperatives–first approach utilizing a customized version of a standard textbook [10]. The course is supported through the publisher's online programming training environment that provides a hands–on learning experience for students, which is supplemented with custom programming homework assignments, and online quizzes in the university's online learning management system. Two consecutive semesters of instruction in the Fall 2014 and the Spring 2015 semesters are used for the evaluation here. The data gathered stems from the course offerings of a single instructor in both semesters under consideration. Several course sections were offered to accommodate the high demand for this course. The same configuration was employed in both semesters, with the laboratory instruction (guided hands–on time) utilizing a computerized classroom with one computer per student.

Students were experiencing different instructional and hands–on times, namely (i) three course sections were aggregated in a large lecture room for two 50 minute lectures per week, supplemented with a single 50 minute guided hands–on instructional sequence at different times depending on the individual section and (ii) a single section was instructed in a smaller classroom for 1.25 hours, supplemented by a 1.25 hour guided hands–on instructional experience. In all instructional lecture components, students were motivated to participate through the usage of in–classroom response systems (commonly referred to as "clickers"). In the earlier Fall semester, a teaching assistant facilitated the hands–on instruction for the sections with a shorter duration, while the course instructor facilitated the hands–on instruction for the longer section. All sections in the Spring 2015 semester were experiencing the same teaching assistant (who was different from the one for Fall 2014) during their hands–on class meeting times.

B. Scoring and Grading

Participation in–lecture was registered and allowed for additional credit. Student submissions of programming assignments and examination programming components were graded by the same graduate teaching assistant in all semesters. The grading graduate student additionally acted as teaching assistant for the short hands–on sessions in the first semester. Students submitted their code through the learning management system in place, where the grading was performed manually employing the rubric presented in Table II. The rubric was shared with students in the beginning of the semesters as part of the instructor's teaching syllabus. In addition to the programming assignments, students were quizzed online using multiple choice/select and true/false question types for each content topic, which aims at reinforcing the students' upkeep with reading and subject knowledge acquisition in a timely manner. Students typically had one week time to complete one or two attempts at a quiz, with typically ten randomly chosen questions from a subset of the textbook's included questions.

Lastly, students were assigned hands–on exercises in addition to the in–person class sessions employing the book publisher's online environment (self–directed hands–on). Students were to answer additional questions as well as submit shorter code fragments, ranging from small entries to several lines of code. The automated system evaluates the code and provides contextual feedback to the learner, who can modify and resubmit the solution an unlimited number of times. Scheduling allows for assignments of deadlines and scores are downloadable to allow latter automated tallying of completed exercises.

C. Major Course Modifications between Offerings

Minor course modifications were made between the semesters with respect to content. The major changes that are considered as part of this paper were the introduction of pass/fail grading paired with tokens for self–empowerment, motivated by recent advocacy, e.g., in [11].

1) Content: Parts of the instructional content sequence were modified as part of general course development for the Spring 2015 semester. The overall course content, however, remained unchanged, i.e., the underlying textbook, slides, and homework training assignments were not varied (but some individual programming assignments were varied with respect to instructional facets). A shift was performed for two mid–semester assignments, which were to program a version of the Tic–Tac–Toe game for a user/computer combination of players. The task was moved towards the end of the semester in the Spring 2015 course offering, which now had an increased difficulty level.

2) Grading: Grading approaches moved to a pass/fail approach, while employing the same rubric for programing assignments. A mapping to a pass/fail was performed by requiring a minimum of (i) 60 % for online open–book quizzes

TABLE II
GRADING RUBRIC EMPLOYED FOR ALL GRADED STUDENT PROGRAMMING ASSIGNMENTS, UNTRIED CRITERIA WERE SCORED AT ZERO POINTS.

Criteria	Attempted	Novice	Competent	Proficient	Excellent
Follows assignment	Compil. errors present, run time errors present, specifics not included, incorrect behavior (2)	Significant details missing, program only works sometimes (4)	Potential for run time errors, does not follow all specifications (6)	Produces incorrect output in some cases (8)	Program is correct in form and function, meets all specifications (10)
Commenting	Almost no comments, no header comment with name, unfitting identifiers, no test at end (1)	Some comments, name present, some errors, whitespace issues/indentation (2)	Average commenting, some errors, identifiers non-intuitive (3)	Most logical steps include comments, name present, test case present, identifiers make sense (4)	Code is well commented, name/tests included, all naming conventions followed, intuitive identifiers (5)
Logic	Logic errors throughout (1)	Incorrect behavior at times (2)	Some small problems in logic (3)	Program gives correct output in some but not all cases (4)	Program works as expected (5)

administered though the learning management system, (ii) 70 % for programming assignments scored employing the rubric presented in Table II, and (iii) 80 % for online hands–on assignments with unlimited trials in order to achieve the passing grade for the item under consideration.

3) Tokens: Students can become more self–empowered if they are enabled to perform their self–scheduling, including the potentials for limited delaying deadlines or skipping of tasks. In turn, setting generous deadlines allows for a gentle introduction of freshmen that have yet to develop their self–scheduling skillset while offering a sensible trade–off with respect to flexibility. To enable additional flexibility, such as skipping an individual task or allowing an extension on a deadline, tokens were introduced into the Spring 2015 course offering. Students were able to earn up to seven tokens from an initial starting balance of one token for specific activities. Not all students decided to perform the required activities to obtain all available tokens. A more in–depth review of the token employment in courses is out of scope here.

III. RESULTS

The overall composition of the course commonly remained unchanged throughout semesters, with N=98 students in Fall 2014 and N=87 students in Spring 2015. Only a small number of students was repeating and significant numbers of students came from across disciplines, allowing the general comparison of results obtained. ANalysis Of VAriance (ANOVA) was applied in some cases where only the normal distribution prerequisite was violated, as ANOVA is considered to be somewhat robust in this case [12], but is omitted for violation of variance homogeneity.

A. Overall Achievement

The initial focus is on the overall student achievement scores attained in both semesters to evaluate the impact of the various course revisions. Overall, students in the 2015 offering performed better on a post-test (M=6.03, SD=1.93) than their counterparts before (M=5.61, SD=3.10). This is corroborated by the total semester score attained as well, where the Fall 2014 students (M=21.98, SD=4.9) were outperformed by the Spring 2015 students (M=23.18, SD=8.29). A Mann-Whitney U test, however, indicated significant differences for this improvement of about 5 % (Mann-Whitney U=5648.5, n1=98, n2=87, $p <$.001). As the addition of tokens as well as pass/fail grading did result in a statistically significant deviation in overall performances, these performances are now assessed in greater detail. Significant differences for the performance in the main number assignments (comparatively eight assignments) and quizzes (comparatively eight quizzes) and are notable. Specifically, the ANOVA for the overall programming assignment performance indicates that the Fall 2014 semester (M=100.34, SD=36.09) performed significantly worse than the Spring 2015 semester (M=116.15, SD=42.21), F(1,183)=7.54, p=.007, η_p^2=.04. Similarly, the student performance increased from M=46.67, SD=14.03 to M=54.42, SD=18.78 in the same time frame, again with significance, F(1,183)=10.24, p=.002, η_p^2 =.053. Overall, significant improvement in the overall course performance can be attributed to the modifications employed, while noting that detailed motivational factors are not part of the evaluation presented here.

B. Guided Hands–On Time

The benefits that can potentially be derived from an increased hands–on instructional component that is delivered in-person is of additional interest. The differences that stem from the course offering in terms of duration of the hands–on exercises, whereby the students in shorter and longer guided hands–on sections are compared. Against intuitive assumptions about the additional personalized training, however, no statistically significant differences for the main components of the course were found (i.e., total scores, programming assignments, or quizzes). For the code understanding component of the final exam in the Spring 2015 semester, however, an ANOVA evaluation revealed that the group with more interactivity time scored significantly higher (M=5.5, SD=2.19) than their counterpart (M=4.12, SD=1.75) on code understanding questions, F(1,74)=7.09, p=.009, η_p^2=.087. A more detailed analysis by semester and group employing a Kruskal–Wallis test indicates that a significant difference between the groups can be observed, with the group experiencing a longer guided hands–on time in the Spring 2015 semester outperforming the next close Spring 2015 short duration group

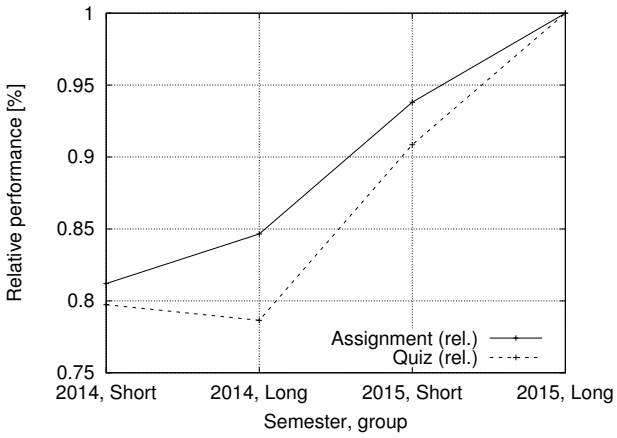

Fig. 1. Average programming assignment totals by semester and guided hands–on instruction.

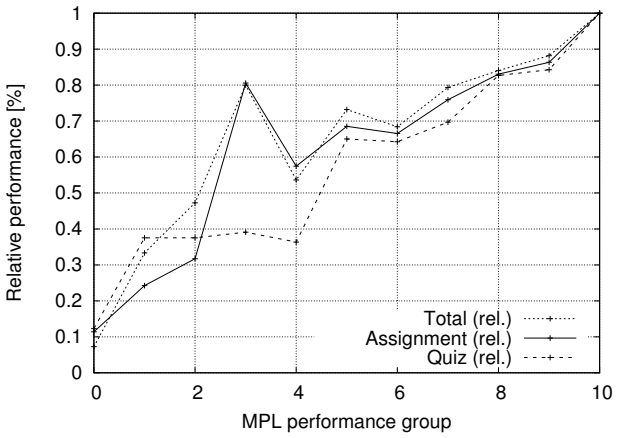

Fig. 2. Average total semester, programming, and quiz scores in relationship to self-directed hands–on performance using MPL.

by 12.5 % (H=17.65, 3 d.f., p=0.001). An ANOVA of the average student performance for programming assignments revealed a significant difference by semester and hands–on time, F(3,181)=2.738, p=.045, η_p^2=.043. The differences in the performance are illustrated in Figure 1, noting that a significant difference was revealed between the short duration in Fall 2014 and the long duration in Spring 2015. The overall course improvements combined with the longer guided hands–on programing time can be seen to have positive impacts on performance. Similarly, the attained scores for quizzes, which exhibited a similar tendency, are illustrated in Figure 1 as well. Again, a significant difference was indicated through ANOVA, F(3,181)=3.898, p=.010, η_p^2=.061, with similar interactions observed for the assignment comparison. Overall, the availability of tokens has likely had the boosting impact on the general level of performance here, as students that otherwise would miss or be unable to perform the required task became enabled to prolong the time on task or skip the task completely. At the level itself, the effects observed in general come into play, i.e., an increased guided hands–on time results in an increased performance on average.

C. Self–Directed Hands–On Performance

In each semester the instruction was supplemented through the publishers online programming hands–on environment; the course offerings employed the Pearson MyProgrammingLab (MPL) online environment that accompanied the selected textbook. The relationship between these self-directed student activities that allow unlimited tries and the potential impact on the resulting semester performance is evaluated next. The performance in MPL is measured based on the relative number of assigned activities that students completed correctly and on time. As these results were distributed across the entire percentage scale, recoding was performed and the student performance was binned in ten percent equi–spaced intervals. Due to inhomogeneous variances, a Kruskal–Wallis test was performed on the different samples, which yielded a significant difference of the distribution of semester total, assignment, and quiz scores, illustrated in Figure 2. For the average total semester score, it can be observed in Figure 2 that a significant (H=97.95, 10 d.f., p <0.001) and almost linear trend between the achievement within the MPL environment and the final total semester scores exists. This indicates that students performing well on the MPL assignments in general will perform better overall. Secondly, the programming assignments, illustrated in Figure 2 as well, exhibit a similarly significant relationship across MPL achievement levels (H=69.19, 10 d.f., p <0.001). Interestingly, these relationships could be seen as almost linear, which is furthermore corroborated through an evaluation of the Spearman's correlation. Indeed, a significant correlation can be observed for all three (note that Pearsons correlation is not significant for the average total semester scores) with correlation coefficients of .277, .299, and .320 for total semester, total assignment, and total quiz scores, respectively.

IV. Conclusion

Course modifications and introducing pass/fail grading with tokens resulted in an observed better learner performance for an introduction to programming course. While some of the gains could be attributed to motivation, student attitudes, or general knowledge levels, results indicate that there are positive effects from the employed modifications. Additional time for guided hands–on exercises increased results for programming assignments as well as quizzes. This hints at potential benefits attainable by "flipping" this particular course. The almost linear relationships of self–guided online training and performance levels in quizzes, assignments, and resulting semester total scores are indicative of the benefits to learners and in line with current research findings. In future works, active learning approaches could be introduced into the evaluation mix to further increase hands–on times, and capturing of student attitudes should provide further insights into underlying mechanisms.

REFERENCES

[1] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming," *SIGCSE Bull.*, vol. 39, no. 2, pp. 32–36, Jun. 2007. [Online]. Available: http://doi.acm.org/10.1145/1272848.1272879

[2] N. Rountree, J. Rountree, A. Robins, and R. Hannah, "Interacting factors that predict success and failure in a cs1 course," *SIGCSE Bull.*, vol. 36, no. 4, pp. 101–104, Jun. 2004. [Online]. Available: http://doi.acm.org/10.1145/1041624.1041669

[3] R. Lister and J. Leaney, "First year programming: Let all the flowers bloom," in *Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20*, ser. ACE '03. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2003, pp. 221–230. [Online]. Available: http://dl.acm.org/citation.cfm?id=858403.858430

[4] L. Porter, C. Bailey Lee, and B. Simon, "Halving fail rates using peer instruction: A study of four computer science courses," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '13. New York, NY, USA: ACM, 2013, pp. 177–182. [Online]. Available: http://doi.acm.org/10.1145/2445196.2445250

[5] M. Sherman, S. Bassil, D. Lipman, N. Tuck, and F. Martin, "Impact of auto-grading on an introductory computing course," *J. Comput. Sci. Coll.*, vol. 28, no. 6, pp. 69–75, Jun. 2013. [Online]. Available: http://dl.acm.org/citation.cfm?id=2460156.2460171

[6] J. Spacco, P. Denny, B. Richards, D. Babcock, D. Hovemeyer, J. Moscola, and R. Duvall, "Analyzing student work patterns using programming exercise data," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '15. New York, NY, USA: ACM, 2015, pp. 18–23. [Online]. Available: http://doi.acm.org/10.1145/2676723.2677297

[7] E. L. Deci, R. Koestner, and R. M. Ryan, "A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation." *Psychological Bulletin*, vol. 125, no. 6, pp. 627–668, Nov. 1999.

[8] G. Leblanc, "Enhancing intrinsic motivation through the use of a token economy," *Essays in Education*, vol. 11, no. 1, pp. 1–20, Aug. 2004.

[9] D. Ariely and K. Wertenbroch, "Procrastination, deadlines, and performance: Self-control by precommitment," *Psychological Science*, vol. 13, no. 3, pp. 219–224, 2002. [Online]. Available: http://pss.sagepub.com/content/13/3/219.abstract

[10] Y. D. Liang, *Intro to Java Programming*. Pearson Education Limited, 2014.

[11] L. B. Nilson, "A better way to grade," *NEA Higher Education Advocate*, vol. 31, no. 5, pp. 6–10, Nov. 2014.

[12] H. J. K. Lisa M. Lix, Joanne C. Keselman, "Consequences of assumption violations revisited: A quantitative review of alternatives to the one-way analysis of variance "f" test," *Review of Educational Research*, vol. 66, no. 4, pp. 579–619, 1996. [Online]. Available: http://www.jstor.org/stable/1170654