# IMPROVED LEARNING THROUGH PEER TUTORING IN A DECLARATIVE PROGRAMMING COURSE

**Abdolhossein Sarrazfadeh and Leon Fourie**
**Unitec Institute of Technology, New Zealand**

**Scott Overmyer**
**Mount Washington College, USA**

## ABSTRACT

*A method of peer tutoring was used and evaluated in an introductory programming course in Haskell. Students were paired on the basis of skill level with pairings opposite those suggested by the existing literature. This method of peer tutoring was shown to be effective in increasing the performance of both high- and low-skilled students in learning a declarative language. This method of peer tutoring can be applied to business and other technology education.*

*Keywords:* Peer tutoring, programming, education, peer learning, improved student performance

## INTRODUCTION

Peer tutoring is defined as "a system of instruction in which learners help each other and learn by teaching", (Goodlad & Hirst, 1989, p. 13). In other words, people of nearly equal states of knowledge teach each other. Peer tutoring takes on a number of forms and is used in a wide variety of disciplines at all levels of education. The authors wanted to know how effective peer learning is in computer science education, since both the efficacy and cost saving potential could contribute to more effective and less expensive computer science education, and since there is a paucity of research on peer tutoring in IT and business higher education.

## PEER TUTORING

Peer learning is expected to have a number of benefits, including better involvement with the material, a stronger level of individual identity as a member of the group of computer science students, prepare students for the group projects that will inevitably come, and expose them to different learning styles and modes of study and problem solving (Wills, Finkel, Gennert & Ward, 1994). These benefits, however, will have to be empirically demonstrated through rigorous, applied research.

Research has also suggested that most novice programmers require special attention and help with programming (Emurian, 2007) and that pairs programming provides novices with a number of benefits in learning languages suggestive of the involvement of some kind of peer tutoring activity during pairs programming (Salleh, Mendes & Grundy, 2011). These benefits extended to general knowledge transfer among pairs, leading us to believe that learning is extended to non-coding tasks as well. In addition, students preferred to pair with students with similar abilities and skills as themselves, suggesting that "peer" tutoring may be

preferred over expert tutoring, where students and teaching assistants or tutors possess significantly differential skill levels (Salleh et al, 2011).

As mentioned in the introduction, peer tutoring has been shown to be an effective method in a variety of educational disciplines. Unfortunately, however, there is very little research on the use of peer tutoring in higher education in business and technology. Peer tutoring has considerable research foundation in K-12 education and , "As one of many teaching techniques available to teach IT, WCPT (peer tutoring) deserves research attention as it applies to IT education." (Cold & Hickman, 2007, p. 51]. Among studies that do exist, one found positive effects on "grade point average, performance rate, success rate and learning strategies and, also, statistically significant pre-post differences for the tutors on learning strategies and social skills." (Arco-Tirado, Fernandez-Martin & Fernandes-Balboa, 2011, p. 773).

Additionally, and very importantly for teaching programming and other IT problem-solving skills is the improvement of student's use of metacognitive knowledge (i.e., the degree to which problem solving infrastructure, such as rules, is instantiated in the student's thinking). In a study of student problem solving … it was determined that peer tutoring had a significant positive effect on the students' level of metacognitive regulation, specifically orienting and planning (De Backer, Van Keer & Valcke, 2012). Metacognitive orienting refers to student's preparation for problem solving by examining goals, task demands, and availing themselves of prior knowledge required solving the problem at hand. "At posttest, students demonstrate significantly more frequent and more varied use of metacognitive regulation, especially during the orientation, monitoring, and evaluation phases. Furthermore, our findings point to an increase in more profound and higher-quality strategy use at posttest." (De Backer et al, 2012, p. 559). This shift in strategy may prove very important in teaching a variety of problem solving skills in IT.

In summary, although research in the use of peer learning in higher education is somewhat sparse, we found that the peer learning literature provides ample evidence to suggest that this is a research avenue worth pursuing.

# HASKELL PEER LEARNING

In order to measure the effect of peer learning on the learning of programming languages in computer science education, an experiment was conducted to assist in identification of misconceptions in students' understanding of programming in order to improve the teaching of the course. The second goal was to measure the improvement in student learning through peer learning. Haskell was chosen partly because some of the authors are involved in the teaching of Haskell, but also because there are clear and distinct concepts in Haskell over which students with a couple of years' experience in more typical languages tend to stumble.

No attempt has been made to determine whether a different set of conceptual blocks would be involved with first time programmers learning Haskell. Most likely orthodox procedural programming is in some ways antagonistic to learning pure functional programming, but this is not the subject of this experiment. The central concern is whether peer learning facilitates the assimilation of Haskell. It is also supposed that most students doing computer science courses will have some interest in orthodox procedural languages, so, at least at tertiary level in computer science, the pristine student may be mythological.

The degree of mathematical sophistication of the students may also affect the learning of Haskell, but it is hard to know exactly how. Some of the concepts can be demonstrated cleanly in Category theory, but students too steeped in maths without experience in programming have at least as much unlearning to do as those coming from procedural languages. The problem here is typically one of program resource usage. However, most of our programming students are not mathematically sophisticated, and it is not our intention in these courses to teach Haskell in that manner. Thus the emphasis is on Haskell as a programming language of a fairly conventional nature, but with some quirks in the syntax and semantics.

# BACKGROUND

The experiment was initially done in a class of 93 students taking declarative programming course in the second academic Semester (offered twice a year at the Albany campus). The course was also offered at the Palmerston North (PN) campus.

Students have historically had difficulty mastering the concepts covered in the course, especially recursion, which forms a substantial portion of the course. Sets of exercises relevant to each concept were given to pairs of students. Pairs were selected based on an exam given at the end of the fourth week of the semester. The literature led us to expect that pairings should be done based on similarities in test scores, however, with this population, the pair consisted of a student who had done well in the exam and one who had had difficulty. We believe that this differential skill level would be more effective with this particular programming paradigm, and that having students of similar knowledge and skills would be either a waste of time (for expert students) or fruitless (for those without baseline Haskell skills). The experiment was aimed at measuring the effect of peer learning and identification of student misconceptions. This paper focuses on the effect of the peer learning experience on the participants learning outcomes.

## The Experiment

Based on the results of a midterm test and assignments given to students in the first half of the semester, pairs of students were selected. Each pair consisted of a student having problems with understanding the concepts (called student A hereafter) and one who had proved mastery with the concept of interest (called student B hereafter). Each set of exercises were used by the pair in the following way:

1. The following steps were repeated for each set of exercises:
   (a) The exercises were given to student A by student B
   (b) Student A solved the problems and student A prepared a report on student A's misconceptions along with the solutions given by A
   (c) Student B then went through each of the exercises with student A trying to help them with the points A had missed
   (d) The same set of exercises was given to A to be solved
   (e) Student B prepared another report on the second solution.
2. Progress of participating pairs was measured by looking at their test and final exam marks. A comparison was also made with the previous term results.
3. The exercises were published as a workbook along with a textbook. The book was bundled with a multimedia intelligent tutor.

# STATISTICAL RESULTS

## Data Overview

A cohort of eighty-two (82) students completed the Declarative Programming course (159.202) at Massey University's Albany Campus in the second semester, of the 93 that had initially enrolled. From this population, 80 had useable data.Following the mid-term test on Haskell programming good students were paired with under-performing students in the peer tutelage programme. A good student was defined as such on the basis of a score not less than 19/30 (63.3%) on the mid-term test. Those receiving a test score less than the threshold were categorised as under-performing. Participation in the programme was determined by the submission of a paired project between a good student and under-performing student. Of the 80 students for whom valid data was available, 42 participated and 38 did not.

As a further indication of the impact that the peer tutoring programme had on student learning, data from the same course run in the first semester of that year was used as a comparison. Suitable data was available from 61 students of the 66 that completed the course in semester 1. This was from an initial class roll of 75. For consistency, student status was classified using the same threshold as that used in the second semester.

The final exam, worth 50% of the student's final mark, had a Haskell component that constituted 28% of the exam and a Prolog component contributing the other 72%.

In order to remove the within-student independence, which is due to the intrinsic properties of the individual student, paired difference data is examined. That is the data obtained from the mid-semester test – collected before the peer tutelage programme commenced – is subtracted from the final mark of the assessment of interest (Haskell or Prolog) obtained after participation for each student. The marks for each type of assessment were converted to a percentage (100%) prior to manipulation enabling relative comparisons. This produced two variables for analyses. Comparisons to the final exam mark are redundant as any influences in the final exam are expressed in the undiluted performance from either the Prolog or Haskell part of the exam.

From a statistical perspective, the data meets the traditional distributional assumptions necessary for parametric inference, namely normality and constant variance. Evidence of this is provided where necessary. The sample size is considered sufficient given the strong fit to the normal distribution. Given that the data is from formal assessment (test and exam) it can be assumed that apart from the influence of the peer tutelage programme results between-student responses are independent. The subtraction of the mid-semester test mark from the relevant exam component negates the within-student dependence.

Prior to commencing an analysis to examine the effect of participation upon final marks, a chi-square test of independence was performed to see if student status (good, under-performing) affected participation in the peer tutelage project of semester two. Under the null hypothesis of independence a p-value of 0.256 was obtained demonstrating that student status did not significantly affect participation. Additionally, a Z-test for differences in proportions indicated no significant difference in the non-completion rates between semesters (p=0.973).

# Methods

Four analyses were performed. The first two were one-way ANOVA to explore the direct impact of student involvement in the peer tutelage programme during the second semester. Where the parametric assumptions failed, a Kruskal-Wallis test was applied. The final two analyses use a general linear model to explore the impact of student involvement whilst incorporating a semester effect. A 5% level of significance is adopted for this study. Normality was tested using Anderson-Darling's method. Where there are only two levels under consideration, an F-test for normal data was used to test equal variances. Bartlett's test for normal data was used to test for equal variances when there were more than two levels.

# Results

Figure 1 shows the impact of the peer tutelage programme upon the relative performance in the exam. Clearly those that participated in the programme performed relatively better in the Haskell component of the exam than their peers.

On average the participants of the programme, with a mean difference of 5.38, performed significantly better (p=0.002) in the Haskell component of the exam than did the non-participants with a mean difference of –6.24. The approximately normal data (p = 0.916 (participants), 0.462 (non-participants)) had equivalent variances (p = 0.08) which are represented by a standard deviation of 18.17 for participants and 13.64 for non-participants.

The relative performance in the Prolog component of the exam, which was not directly involved in the peer tutelage project, is displayed in Figure 2 by level of participation. There is no clear distinction between these two groups. No significant difference exists (p=0.532) in the relative performance of the Prolog component of the exam. The data was approximately normal for both the participants ($\sim N(-15.63, 19.30^2)$ p=(0.523)) and the non-participants ($\sim N(-12.99, 18.29^2)$ (p=0.597)).

The students from the second semester cohort were broken into four categories; non-participating under-performing students (US NP), non-participating good students (GS NP), participating under-performing students (US P) and participating good students (GS P) to see which group benefited the most from involvement in the programme.

**TABLE 1. PROPERTIES OF THE RELATIVE HASKELL MIDTERM DATA FROM THE SECOND SEMESTER**

| Level | n | Mean | Standard Deviation | Test for ~N (p) |
|---|---|---|---|---|
| **UP NP** | 21 | -1.39 | 12.45 | 0.624 |
| *G NP* | 17 | -12.24 | 12.94 | 0.697 |
| *UP P* | 20 | 11.74 | 22.21 | 0.832 |
| *G P* | 22 | -0.40 | 11.15 | 0.399 |

There is a significant difference between the mean relative performances of the groups. The data is approximately normal, as shown by the p-values from the Anderson-Darling normality test in Table 1. However, the variances are not equivalent (p=0.006). A Kruskal-Wallis test reinforces the previous result that there is a significant difference between groups (p=0.001).

**Figure 1. Boxplot for Relative Performance in Haskell by Level of Participation**
**FIGURE 2. BOXPLOT FOR RELATIVE PERFORMANCE IN PROLOG BY LEVEL OF PARTICIPATION**

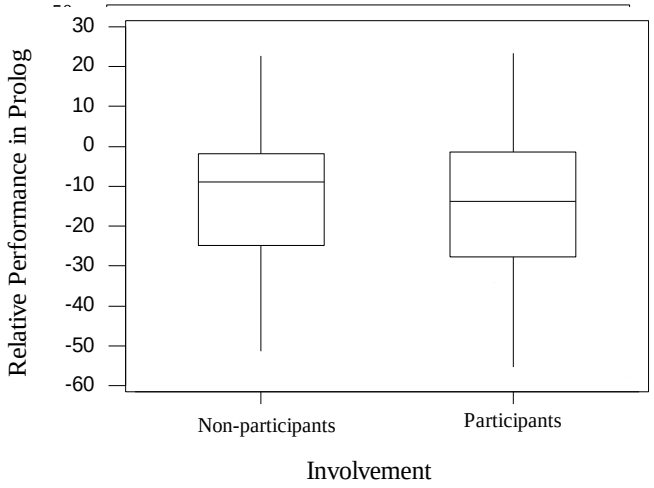
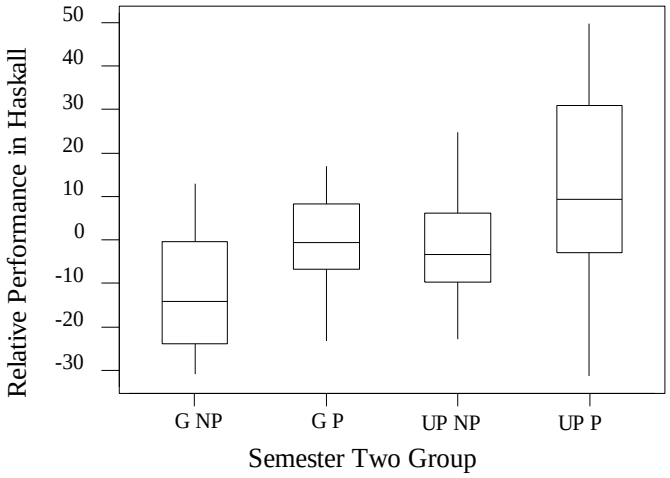



**Figure 3. Boxplot for Relative Performance in Haskell by Group**




A general linear model (GLM) was used to explore the impact of participation while accounting for a semester effect. As semester one did not have a peer tutoring programme running, this cohort provides a

benchmark to evaluate the relative performance in Haskell and determine if there was any undue influence (deliberate or accidental) affecting performance.

The general linear model adopted is of the form:

$$E(y) = \mu + Se + In + St + In \times St,$$

where E(y) is the relative performance in the exam portion of interest, $\mu$ is the constant of the process (overall mean relative difference), 'Se' is the semester effect(one/two), 'In' is the involvement effect (participant/non-participant), 'St' is the status of the student (good/under-performing) and 'In × St' is the interaction between involvement and student status.

To employ the GLM successfully we assume that the data for the different levels of each variable is approximately normal with equivalent variance. This is the case as shown in Tables 2 and 3.

The general linear model for the relative performance on Haskell has two significant terms, involvement (p=0.001) and status (p=0.001), as shown in Table 4. All other terms are deemed insignificant, that is the coefficients are not significantly different from zero. The residuals were approximately normal (0.00, $19.24^2$) (p=0.114) and exhibit homogeneity of variance and no correlation when plotted against the fitted values.

The second general liner model examining the influences upon the relative performance on Prolog has only one significant term (excluding the constant), a semester effect (0.005). The residuals were approximately normal (0.00, $15.75^2$) (p=0.357) and exhibit homogeneity of variance and no correlation when plotted against the fitted values.

**TABLE 2. PROPERTIES OF THE RELATIVE HASKALL - MIDTERM DATA FROM BOTH SEMESTERS**

| Variable | Level | Mean | Standard Deviation | n | Test for ~N (p) | Constant Variance (p) |
|---|---|---|---|---|---|---|
| **Semester** | 1 | -11.08 | 17.04 | 61 | 0.388 | 0.986 |
| | 2 | -0.14 | 17.10 | 80 | 0.140 | |
| *Involvement* | NP | -9.22 | 15.93 | 99 | 0.490 | 0.294 |
| | P | 5.38 | 18.16 | 42 | 0.246 | |
| *Status* | G | -9.37 | 15.79 | 71 | 0.907 | 0.151 |
| | UP | -0.30 | 18.78 | 70 | 0.065 | |

**TABLE 3. PROPERTIES OF THE RELATIVE PROLOG-MIDTERM DATA FROM BOTH SEMESTERS**

| Variable | Level | Mean | Standard Deviation | n | Test for ~N (p) | Constant Variance (p) |
|---|---|---|---|---|---|---|
| *Semester* | 1 | -1.94 | 20.99 | 61 | 0.624 | 0.346 |
| | 2 | -14.38 | 18.75 | 80 | 0.245 | |
| *Involvement* | NP | -6.18 | 20.62 | 99 | 0.964 | 0.645 |
| | P | -15.64 | 19.30 | 42 | 0.596 | |
| **Status** | G | -12.80 | 20.71 | 71 | 0.447 | 0.757 |
| | UP | -5.15 | 19.95 | 70 | 0.914 | |

**Table 4. GLM results for the difference between Haskall exam percentage and midterm test percentage per student**

| Source | Associated Level | Coefficient | S.E. Coefficient | P-Value |
|---|---|---|---|---|
| **Constant** | | -2.604 | 1.604 | 0.107 |
| *Semester* | 1 | -2.108 | 1.656 | 0.205 |
| *Involvement* | NP | -6.168 | 1.792 | 0.001 |
| *Status* | G | -5.027 | 1.474 | 0.001 |

| | | | | |
|---|---|---|---|---|
| *Involvement×Status* | NP×G | 1.038 | 1.474 | 0.482 |

**Table 5. GLM results for the difference between Prolog exam and midterm test percentage per student**

| Source | Associated Level | Coefficient | S.E. Coefficient | P-Value |
|---|---|---|---|---|
| **Constant** | | -2.604 | 1.604 | 0.000 |
| *Semester* | 1 | -2.108 | 1.656 | 0.005 |
| *Involvement* | NP | -6.168 | 1.792 | 0.643 |
| *Status* | G | -5.027 | 1.474 | 0.025 |
| *Involvement×Status* | NP×G | 1.038 | 1.474 | 0.933 |

# Discussion

There is significant evidence that the peer tutelage programme is beneficial to those who participate. The results from the ANOVA support this as there is a significant difference in the relative performance on the Haskell component of the exam by level of involvement, but there was no significant distinction in the Prolog component.

In the second semester, the under-performing participating students benefit the most from involvement in the programme, improving on average by 11.74% on the mid-term percentage in the Haskell part of the exam compared to their peers. Under-performing students who did not participate continued on average to underperform. That is the mean (–1.39) for this group indicates no significant relative improvement had been made. A similar situation applied to the good students who participated. On average they made no significant gains (-0.40). However, this signifies that on average they remained "good students". Collectively, the good students who did not participate fared comparatively worse with respect to their peers. This group's relative performance (–12.24) in the Haskell component of the exam was significantly worse than that in the mid-term.

The general linear model that considers both semesters reinforces the results from the second semester. The first model for the relative performance in the Haskell part of the exam has two significant variables, student status and participation.

Importantly, participation results in a relative increase of 6.2% over non-participants. Additionally, under-performing students benefit by 5% more than the good students do. As the interaction between student status and project participation is not significant the benefits of the peer tutelage programme are equivalent for good and under-performing students.

Further, because the semester effect is not significant, this indicates that the peer tutelage programme has significant benefits to those who participated and these benefits were not caused by an increased emphasis in teaching Haskell or changes in teaching this part of the course between semesters.

The second general linear model, which examined relative performance in Prolog, also supports the previous findings. Participation is not significant in this model (p=0.643), suggesting that relative improvements in Haskell were a result of the peer tutelage programme. Whilst students in the first semester performed relatively better on the Prolog component of the course, this is not due to the peer tutelage programme, but due to any number of unknown causes such as the difficulty of the exam questions, delivery of material and so forth. This semester effect would only have been important if involvement had also been significant in this model. This would have implied that students or staff in the tutelage programme had placed more emphasis on Haskell to the detriment of learning Prolog.

The results from the two semesters of the Declarative Programming course at Massey University's Albany Campus show conclusively that the peer tutelage programme was beneficial to the participants' relative performance in the Haskell component of the exam.

# CONCLUSION

Overall, our experiment shows that peer tutoring can be beneficial to teaching programming, as was our expectation and hypothesis. What is slightly counterintuitive is the notion that peers of differential skill and knowledge can be effectively paired in certain circumstances. We did not expect this effect based on our expectations from the literature, which suggests that pairs of like ability result in both the preferred pairings, as well as the most effective pairings form a peer tutoring standpoint (Salleh et al, 2011; Han et al, 2010; Sklar et al, 2006). As it turned out, the interaction between student status and project participation is not significant, suggesting that the benefits of the peer tutelage programme are equivalent for good and under-performing students using a reciprocal pair tutoring (RPT) paradigm.

Based on both our experiments and the literature, we can make the following recommendations:

1) Peer tutoring should follow a paradigm, and should have a method in place for best effect.
2) Peer tutoring was used after initial learning, to determine the knowledge and skill differential between students. This timing is supported in the literature and is it suggested that peer tutoring best serves students for whom traditional learning has not been maximally successful (Diosos-Henson, 2012).
3) In addition to learning the Haskell language, students were also observed to have learned other skills such as communication, collaboration, planning and other related IT problem solving skills.
4) Students performed well in a RPT paradigm in our study, and we can recommend this paradigm as effective for both peers. This view is also supported in the literature (Dioso-Henson, 2012).

One next possible step is to repeat this study using a peer matching paradigm, in which students of equal skill levels are paired, and the data is compared to results from other studies, including the current study. This comparison would tell us if the matching paradigm employed is significant as suggested by the literature, or whether the matching paradigm might be task dependent.

# REFERENCES


Arco-Tirado, J. L., Fernandez-Martın, F. D., & Fernandez-Balboa, J-M. (2011) "The impact of a peer-tutoring program on quality standards in higher education", *Higher Education: The International Journal of Higher Education and Educational Planning*, *62*(*6*), 773-788.

Cold, S. J. & Hickman G. D. (2007) "Literature review and experience with whole classroom peer tutoring for IT students". In *Proceedings of the 8th ACM SIGITE conference on Information technology education* (SIGITE '07). ACM, New York, NY, USA, 49-52.

De Backer, L., Van Keer, H., & Valcke M. (2012) "Exploring the potential impact of reciprocal peer tutoring on higher education students' metacognitive knowledge and regulation", *Instructional Science: An International Journal of the Learning Sciences*, *40*(*3*), 559-588.

Dioso-Henson, L. (2012) "The effect of reciprocal peer tutoring and non-reciprocal peer tutoring on the performance of students in college physics", *Research in Education*, 87(1), 34-49.

Emurian, H. (2007) "Teaching Java: Managing instructional tactics to optimize student learning", *International Journal of Information and Communication Technology Education*, *3*(*4*), 34-49.

Goodlad, S., and Hirst, B. (1989) *Peer Tutoring: A Guide to Learning by Teaching.* New York: Nichols Publishing.

Han, K., Lee, E., & Lee, Y. (2010) "The Impact of a Peer-Learning Agent Based on Pair Programming in a Programming Course", *IEEE Transactions on Education*, 53(2), 318-327.

Salleh, N., Mendes, E., & Grundy, J (2011) "Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review", *IEEE Transactions on Software Engineering*, *37*(*4*), 509-525.

Sklar, E. and Richards, D. (2006) "The use of agents in human learning systems". In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems* (AAMAS '06). ACM, New York, NY, USA, 767-774.

Wills, C., Finkel, D., Gennert, M., and Ward, M. (1994) "Peer learning in an introductory computer science course". In: *Proceedings of the Twenty-Fifth SIGCSE Symposium on Computer Science Education* (SIGCSE '94). ACM, New York, NY, USA, 309-313.