Simple Feedback Can Do the Job

Analyzing the Effects of Simple Computer Based Feedback for Fundamental Programming Tasks

Matthias Laengrich
Faculty of Electrical Engineering and Computer Science
University of Applied Sciences Zittau / Goerlitz, Germany
m.laengrich@hszg.de

***Abstract*—Although, the importance of Computer Science has been increasing, it must be noted that the drop-out rate among students of STEM subjects unfortunately is among the highest we know in education, worsen the fact that the enrollment rates are relatively small. Thus, we might have to look closer at pedagogical approaches to overcome this problem. The New Zealand educator John Hattie introduced a meta-study called "Visible Learning" in which he mentions feedback to be one of the major factors affecting the success of learning. However, its influence on fundamental programming tasks cannot be investigated properly unless objective, reliable and valid tasks exist that allow a clear inference of the observed performance for the given feedback. For example it would be possible to argue that observed educational "improvements" are in fact the result of a reliability or validity problem. Since tasks for programming fundamentals were often created intuitively, it had to be explored how to get objective, reliable and valid tasks. After their development, an empirical study has shown that simple feedback can indeed make a contribution to the successful learning of programming fundamentals if the corresponding tasks are less complex according to Anderson's taxonomy.**

Keywords—Engineering Education

I. INTRODUCTION

Teaching concepts of Computer Science, in particular the fundamentals of programming, is a major challenge and often underestimated by students as well as lecturers [1,2]. This leads to small enrollments and to the highest drop-outs compared with other fields of studies at universities also [3]. The fact that CS concepts are more and more taught at courses that focus on other learning subjects (such as Mechanical Engineering) are due to the fact that the importance of CS increases permanently [4]. Despite this positive development the problem the lecturers are facing now is even more problematic: an even less motivated audience is being presented with learning objectives (e.g. programming fundamentals) that are already challenging to CS students but even more to others. So how can we overcome this challenge that seems to be one of the hardest we're faced with in education?

One very interesting approach has been presented by the New Zealand educator John Hattie that might give this ongoing discussion a new strong impulse [5]: the positive effects of feedback. "There is a preponderance of evidence that feedback is a powerful influence in the development of learning outcomes. Two findings from the many meta-analyses of the effects of feedback are most fascinating—the average effects of feedback are among the highest we know in education, and feedback effects are among the most variable in their influences" [6]. Thus the question arises how to use this positive effects for the learning of programming fundamentals. This paper discusses a study that has investigated simple computer based feedback (right/wrong) based on tasks that were not developed intuitively but strongly theory driven. The study results indicate that simple computer based feedback won't be a good choice for all kinds of tasks but for less difficult ones.

II. FEEDBACK AS SEEN FROM FUNDAMENTAL PROGRAMMING TASKS

Feedback (to feed back), as Hattie is using the term, means an discrete information provided for the student after or during solving a task. In case the student didn't solve a task completely or correct he/she will be informed about that fact by the computer [7]. The feedback source may be the tutor or other peers and is, in most cases, considered to be very expensive (time consuming).

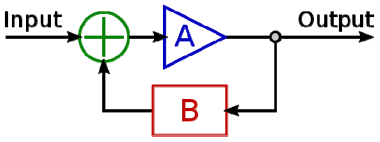

Fig. 1. A typical feedback-loop. Source: Wikimedia Commons

[8] states that a success is being attributed internally by the student (I solved it myself!), thus, strongly strengthen the students motivation (I know how to do it!). In case of negative feedback the student has to overthink his/her solution approach, hopefully finding a way and remembering it permanently (learning). In case no feedback is being provided at all the success of learning might be very questioning [9]. However, results of [10,11] show that even simple or worse feedback is better than nothing. To provide feedback to the students is very expensive because many classes rely on just one tutor. That's why it may be a good approach to support the tutor by developing a computer based learning environment

[12]. However, the feedback quality of a learning environment is not always as good as human tutor feedback [13,14]. As a workaround the students may ask each other but peer feedback is already known to be erroneous and contradictory [15].

Under these circumstances: Can simple computer based feedback be of any help by the learning of the fundamentals of programming? Its big advantage may be that its development is quite cheap. But this advantage only counts in case it has a positive effect at all. To answer this question an empirical study is needed and, considering many important design details, comparing the performance of two randomized groups solving the same tasks with one group receiving feedback and a control group that doesn't in a pretest-treatment-posttest-setting. In case a performance difference can be observed the given feedback must be responsible in case other variables can be eliminated.

Unfortunately, [16] found that tasks for programming fundamentals were often developed intuitively and not theory driven. This has the consequence that the results of these tests cannot be evaluated in an objective way but leading to different evaluations (different points, different opinions concerning the completeness) depending on the evaluator. Non-objective tasks are also not reliable. The task's validity hasn't been proofed but was assumed only. These intuitive tasks may be very important and useful from a didactical perspective. From an empirical, however, their suitability is questionable.

It's important to be aware of the fact that tasks that should be used as an empirical measurement instrument for the operationalization of skills, must meet the demands of every ordinary empirical measurement instrument, such as objectivity, reliability and validity at least [16,17].

III. How to ensure Objectivity, Reliability and Validity of fundamental programming tasks

Apart from other quality criteria the objectivity, reliability and validity are considered the most important ones in this setting [18]. [16,17,19] are concerned with the question of how the objectivity, reliability and validity of fundamental programming tasks can be guaranteed. Without this basic question answered a further study of the effectiveness of simple computer based feedback would probably lead to unreliable results.

Objectivity relates to ambiguities in the assessment of the completeness and correctness of a result. In order to attain objectivity [19] suggests to develop a formalized task description format. It can be stated that many tasks related to their specific description formats are very similar. This can lead to the development of task types that work similar to superclasses. Their description is the same for all tasks of this type.

Additional reliability problems arise due to the fact that the learners understand one and the same task differently and, as a consequence, develop different solution approaches. Again, this is due to an ambiguous interpretation of the task and can be achieved by formalizing the tasks description (see. this particular [16]).

Deciding, whether a task is valid or not is indeed often assumed but not proofed hard enough. [16] demonstrates that this question can be answered following a strict procedure model when developing tasks.

IV. Development of tasks

Based on the above-mentioned work, tasks were developed that meet the quality criteria of objectivity, reliability and validity. Following the recommendations of [16,19] two types of tasks have been developed with varying degrees of difficulty, which are presented below.

TABLE I. Description of the Task Type „Aggregated Assignment".

Property	Details
Name	Aggregated Assignment
Abbreviation	AA
Skills adressed	Understanding post-increment expressions, post-decrement expressions and aggregated assignments
Difficulty	Level 2 (understand) [a]
Given	Post-increment expressions, post-decrement expressions and aggregated assignments.
Main operator	Note the equivalent simple assignment, a declaration as well as an initialization for all input and output variables.
Solution representation	Table with 5 columns: (1) No, (2) equivalent simple assignment, (3) declarations of all variables, (4) initializations of output variables.
Assessment	Every correct cell one point.

[a.] According to Anderson and Krathwohl [20].

Since the task type description already contains a lot of information the corresponding tasks can be very short and limited to the actual assignments.

```
(0) x |= y;
(1) g -= 2*a*b;
```

TABLE II. Solution of the two AA-tasks.

No	1 (simple)	2 (decl.)	3 (init. all) [b]	4 (init. out)
0	x = x | y;[c]	bool x, y;	x = true; y = false;	x = true;

1	g = g – (2*a*b)	int a, b, g;	a = 1;
			b = 2;
			g = 3;
				g = -1;

[b.] It's up to the student how to initialize the variables as long as the chosen values are valid and the calculated output is correct. The solution isn't available for the students.

[c.] The expression x=y|x; would be correct as well of course.

The solutions are in no case available to the student's since [8] shows that the student would otherwise not associate the success with his/her own performance but with the fact that the solution was already there. Thus, not providing a solution strengthens the student's motivation and self-confidence. In case the solution is already available the student may not be motivated to perform at all [21].

The second task that has been used during the study was an implementation tasks with the highest difficulty of 6 (create) according to Anderson's taxonomy. Its task type description follows.

TABLE III. DESCRIPTION OF THE TASK TYPE "IMPLEMENTATION".

Property	Details
Name	Implementation
Abbreviation	I
Skills adressed	Understanding the implementation of algorithms using methods.
Difficulty	Level 6 (create)[e]
Given	Value ranges of variables, requirements / limitations, specification[d].
Main operator	Realize the algorithm.
Solution representation	List of statements placed in a method's body.
Assessment	Every successfully tested aspect of the specification one point.

[d] In case no algorithm to be implemented is provided so the students will have to find them by themselves. Otherwise the difficulty drops to 3 (apply).

[e] The specification lists a number of aspects to be taken into consideration. These aspects can be reused for assessment purposes.

A valid task following this description has been developed and is shown below.

Value Ranges

In: bool t1, t2, a1; int p;

Out: string state;

Specification[1]: On "state" calculate the student's state after absolving the exam. The following conditions apply:

- If t1 (test 1) is false (not passed) decrement the student's points "p" by one, otherwise nothing is changed.
- If t2 is false decrement the student's points by 2, otherwise nothing is changed.
- If a1 (additional task) is true and the student's points are less or equal to 13 then they are incremented by 2.
- In case p is less than 5 the student's state is "not passed".
- In case p is equal or greater than 5 the student's state is „passed".

Solution[2]

```
0   if( !t1 )
1   {
2     p--;
3   }
4   if( !t2 )
5   {
6     p -= 2;
7   }
8   if( a1 && p<=13 )
9   {
10    p += 2;
11  }
12  if( p < 5 )
13  {
14    state = "not passed";
15  }
16  else
17  {
18    state = "passed";
19  }
```

With these two tasks a study was prepared to investigate the effect of simple computer based feedback. The students had to solve the given tasks in a learning environment providing a check button to check the correctness of the entered solution any time and as often as they want.

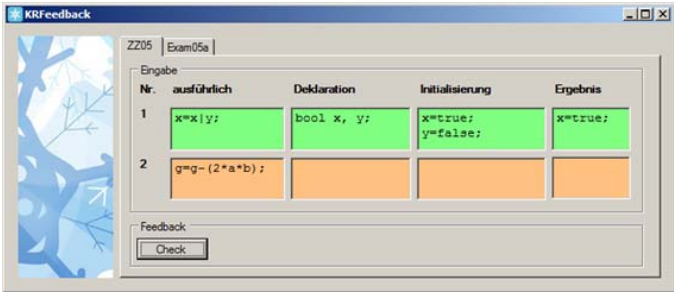

Fig. 2. Screenshot of the learning environment showing the AA tasks with the first expression already green and the second incomplete yet.

V. STUDY RESULTS

The study's hypothesis is that simple computer based feedback (right/wrong) of fundamental programming tasks provided by a computer based learning environment helps students in a significant way to succeed solving these tasks as if they wouldn't receive any feedback (control group)[3].

A typical design approach has been chosen in which 40 first semester students of Mechanical Engineering were randomly assigned to one of two groups. The study itself took place in three phases starting with a pretest containing two tasks of the above mentioned types. No group received any feedback. This information was later used to measure the students previous knowledge, which has to be taken into consideration according to [23]. Students solving the tasks completely right do not need any feedback at all and their data was ignored during the following discussion.

The pretest was followed by the treatment where the control group received no feedback (again) and the other group received simple computer based feedback. Again the students were confronted with two tasks of the above mentioned types.

Finally, a posttest took place providing no feedback to both groups again as it was practiced during the pretest. The aim of the posttest was to find out whether or not the students were able to permanently show a progress in their learning outcomes, thus, a better learning performance compared to the control group.

[1] This specification lists 5 aspects to consider during the assessment of this task.

[2] Again, the solution isn't provided to the students. Apart from that multiple solutions may exist so this approach can be seen as an example.

[3] The study's results were successfully presented to a German audience only so far in [22]. Its, therefore, the wish of the author to address the English speaking community now.

The results found by this study were quite surprising. Since [5] stated that feedback will have a positive impact and [10,11] found that any feedback might be better than no feedback it was assumed that the feedback group will have no problems in outperforming the control group. However, this wasn't always the case. The discussion of the study's results starts with the AA-task (level 2, understand) followed by the more difficult I-task (level 6, create). The following table shows to what degree (percent) the groups were able to solve the AA-task during all phases of the study.

TABLE IV. COMPARISON OF THE GROUPS AA-TASK PERFOMANCE.

Phase	Feedback group n=13	Control group n=13	p-value
Pretest	mean .44 stddev .34	mean .51 stddev .25	p .567
Treatment	mean .71 stddev .34	mean .59 stddev .30	p .330
Posttest	mean .74 stddev .36	mean .59 stddev .32	p .233

As the p-value indicates none of the groups were able to perform better than the other in a statistically significant way. However, this result may be distorted by the fact that the control group started with a mean value of .51 and a quite low stddev (.25) outperforming the feedback group already at the beginning. While the stddev remains almost the same during the study the feedback group was able to increase its performance up to a mean of .74. However, the t-test wouldn't consider this increase as statistically significant.

Another picture on the same data is drawn by the index introduced by Phye and Bender [24], the so called corrective probability index. This index seems to be more interesting under these circumstances because it focuses on the probability the students were able to fix an error that occurred during the pretest later on in the posttest. The potential for doing so is greater for the feedback group because they started with a lower mean value, thus, with more errors probably to be fixed later on.

$$CP = \frac{w_1 \rightarrow r_2}{(w_1 \rightarrow r_2) + (w_1 \rightarrow w_2)} \quad (1)$$

Phye and Bender calculate the corrective probability by counting all errors of the pretest (w_1) that have been corrected in the posttest (r_2) divided by the sum of corrected and incorrected errors. Considering the CP the following picture applies.

TABLE V. CONSIDERING THE CORRECTIVE PROBABILITY THE FEEDBACK GROUP OUTPERFORMED THE CONTROL GROUP

	Feedback group n = 13	Control group n = 13	p-value
CP	mean .61 stddev .45	mean .26 stddev .38	p .04

It was observed that the feedback group was indeed able to correct errors that were made during the pretest with a probability of .61 in the posttest. The control group only achieved a probability of .26 with a stddev .07 below the feedback group. It may, therefore, be assumed that simple computer based feedback, which was the control variable, can help supporting the students success of learning with these kinds of tasks. However, since the n of 13 is quite small further studies are necessary to increase the level of reliability.

With these outcomes in mind the results of the second task, which was the complex I-task (level 6, create), were analyzed as well. Again, it was assumed that simple computer based feedback would make a contribution to the success of learning.

TABLE VI. COMPARISON OF THE GROUPS I-TASK PERFOMANCE.

Phase	Feedback group n=14	Control group n=14	p-value
Pretest	mean .41 stddev .36	mean .46 stddev .35	p .733
Treatment	mean .45 stddev .30	mean .49 stddev .16	p .812
Posttest	mean .37 stddev .31	mean .43 stddev .30	p .625

As the results show the feedback group wasn't able to outperform the control group in any phase. Another look at the above mentioned corrective probability doesn't change the picture.

TABLE VII. THE CORRECTIVE PROBABILITIES DO NOT DIFFER IN A RELEVANT WAY CONSIDERING THE I-TASK.

	Feedback group n = 14	Control group n = 14	p-value
CP	mean .08 stddev .12	mean .04 stddev .07	p .368

These results differ completely compared to those discussed for the AA-tasks and rise a complete new discussion. Before the study it was known, due to the research of [25], that feedback studies seem to contradict each other by sometimes identifying an effective feedback that doesn't work in another setup. It was, up to then, unclear why the results presented by various studies didn't always show the same effects of the same kind of feedback. This study may present another important variable that probably hasn't been considered enough in the past: the task's difficulty.

Thanks to not developing the tasks intuitively but theoretically based it can be proofed without a doubt that the both task types presented here have a different difficulty, can be assessed in an objective way and leading to reliable results. Knowing that simple computer based feedback may not be appropriate for difficult task types is important to know since it may help lecturers not relying on learning environments offering this kind of feedback to their students. However, the results are quite promising for less difficult tasks it is strongly recommended to improve this study's reliability due to its small n.

VI. CONCLUSIONS

The main advantage of using simple computer based feedback may be seen in its cheap price compared to human feedback. However, its effects may not apply to all task types used during a study course. That this is the case has only been discovered because of the fact that the difficulty of the tasks used during the study had been clearly described before. Thus,

simple computer based feedback may be successful for less difficult tasks it remains open how to handle difficult tasks with the help of learning environments.

VII. Acknowledgments

The author would like to thank Prof. Dr. Azizi Ghanbari, Prof. Dr. (em.) Schott and Prof. Dr. Schulze for their continuous and helpful support as well as the anonymous reviewers, who gave valuable feedback for further improvement of this paper.

VIII. References

[1] Lister, Raymond. ed. 2007. *Koli Calling 2007. Proc. Seventh Baltic Sea Conference on Computing Education Research.* Darlinghurst: Australian Computer Society. (Conferences in research and practice in information technology series, vol. 88).

[2] Oliver, Dave, Tony Dobele, Myles Greber, and Tim Roberts. "This course has a Bloom Rating of 3.9." In R. Lister and A. Young (eds). 2004, *Proceedings of the Sixth Australasian Conference on Computing Education*. Darlinghurst: Australian Computer Society, Inc. (Volume 30), pp. 227–231.

[3] Heublein, Ulrich, Johanna Richter, Robert Schmelzer, and Dieter Sommer. 2014. "Die Entwicklung der Studienabbruchquoten an den deutschen Hochschulen. Statistische Berechnungen auf der Basis des Absolventenjahrgangs 2012." http://www.dzhw.eu/pdf/pub_fh/fh-201404.pdf. Accessed: 12 February 2015.

[4] ACM Education Policy Committee. 2014. *Rebooting the Pathway to Success. Preparing Students for Computing Workforce Needs in the United States.* New York: ACM.

[5] Hattie, John. 2009. *Visible learning. A synthesis of over 800 meta-analyses relating to achievement.* London, New York: Routledge.

[6] Hattie, John A. and M. Gan. "Instruction based on Feedback." In R. E. Mayer and P. A. Alexander (eds). 2011, *Handbook of research on learning and instruction*. New York: Routledge (Educational psychology handbook series), pp. 249–271.

[7] Horowitz, Paul and Winfield Hill. 1981. *The Art of electronics.* Cambridge: Univ. Pr.

[8] Narciss, S. 2013. "Designing and Evaluating Tutoring Feedback Strategies for digital learning environments on the basis of the Interactive Tutoring Feedback Model." *Digital Education Review*. Vol. 23, pp. 7–26.

[9] Skinner, B. F. "The science of learning and the art of teaching." In R. A. Patton (ed.). 1954, *Current trends in psychology and the behavioral sciences*. Pittsburgh and PA: University of Pittsburgh Press , pp. 38–58.

[10] Azevedo, R. and R. M. Bernard. 1995. "A meta-analysis of the effects of feedback in computer-based instruction." *Journal of Educational Computing Research*. 13(2), pp. 111–127.

[11] Clariana, R. B. 1993. "A review of multiple-try feedback in traditional and computer-based instruction." *Journal of Computer-Based Instruction*. 20(3), pp. 67–74.

[12] Längrich, Matthias and Jörg Schulze. "A Systematic Approach to Immediate Verifiable Exercises in Undergraduate Programming Courses." 10/28-10/31/2006. *Proceedings of the 36th Annual Frontiers in Education Conference*. Piscataway, NJ: Institute of Electrical and Electronics Engineers (IEEE), pp. 1112–1116.

[13] Merrill, D. C., Reiser, B. J., Merrill, S. K., and S. Landes. 1995. "Tutoring: Guided learning by doing." *Cognition and Instruction*. Vol. 13 (3), pp. 315–372.

[14] Corbett, A. T., Anderson, J. R., Graesser, A. C., K. Koedinger, and K. VanLehn. eds. 1999. *Third generation computer tutors: Leam from or ignore human tutors?* New York: ACM Press.

[15] Hamer, John, Helen Purchase, Andrew Luxton-Reilly, and Paul Denny. 2014. "A comparison of peer and tutor feedback." *Assessment & Evaluation in Higher Education*. Vol. 40 (1), pp. 151–164.

[16] Längrich, Matthias and Jörg Schulze. "Rethinking Task Types for Novice Programmers." 10/23/2014-10/25/2014. *Proceedings of the Frontiers in Education Conference*. Piscataway, NJ: Institute of Electrical and Electronics Engineers (IEEE), pp. 2597–2604.

[17] Längrich, Matthias, Jörg Schulze, and Amruth N. Kumar. "Expression Tasks for Novice Programmers. Turning the Attention to Objectivity, Reliability and Validity." 10/21/2015-10/24/2015. *Proceedings of the Frontiers in Education Conference*. Piscataway, NJ: Institute of Electrical and Electronics Engineers (IEEE), pp. 300–307.

[18] Bortz, J. and N. Döring. 2002. *Forschungsmethoden und Evaluation.* 2nd ed. Heidelberg: Springer.

[19] Längrich, Matthias, Jörg Schulze, and Shahram Azizi Ghanbari. 2013. "Anwendung eines allgemeinen Aufgabenbeschreibungsformates auf die Imperative Programmierung." *grkg Humankybernetik*. 54(2), pp. 64–76.

[20] Anderson, Lorin W. and David R. Krathwohl. 2001. *A taxonomy for learning, teaching, and assessing. A revision of Bloom's taxonomy of educational objectives.* New York: Longman.

[21] Anderson, Richard C., Raymond W. Kulhavy, and Thomas Andre. 1971. "Feedback procedures in programmed instruction." *Journal of Educational Psychology*. Vol. 62 (2), pp. 148–156.

[22] Längrich, Matthias. 2016. Aufgaben und Feedback. Pädagogische Herausforderungen der Grundlagen-Programmierausbildung. Dissertation. TU Dresden.

[23] Zielinski, Werner. 1998. *Lernschwierigkeiten. Ursachen - Diagnostik - Intervention.* 3rd ed. Stuttgart, Berlin, Köln: Kohlhammer. (Kohlhammer-Standards Psychologie).

[24] Phye, G. D. and T. Bender. 1989. "Feedback complexity and practice: Response pattern analysis in retention and transfer." *Contemporary Educational Psychology*. Vol. 14, pp. 97–110.

[25] Narciss, Sunanne. 2006. *Informatives tutorielles Feedback.* Münster: Waxmann.