Self-Efficacy in Programming among STS Students

Kasper Davidsson, Lars-Åke Larzon, Karl Ljunggren

Abstract

A few years ago, the introductory programming course for STS students, engineering students within sociotechnical systems engineering, at Uppsala University was redesigned according to principles presented in [5]. One concern that was raised after this change is whether those students are now appropriately prepared for future programming courses with the new design in the introduction course. To investigate this closer, a starting point is to study the students' self-efficacy beliefs, i.e., their judgment about their own programming capabilities as there are other studies that confirms a correlation between students self-efficacy in programming and their success in learning and using programming. [7, 8]

We have investigated how the self-efficacy in programming among STS students change during the year after their introductory programming course and have also studied how well they consider themselves to meet specific course goals from the programming course. Our results show no significant difference in the overall self-efficacy score, but a significant increase in self-regulation. Students perceive an increase in most of the abilities expressed by course goals investigated. We find no indication that students would be insufficiently prepared to face programming tasks in other courses after their introductory course on the topic.

Introduction and Motivation

Self-efficacy theory is used within several domains concerning education, training or other activities where a person is to attain a new or develop a higher level of skill. The theory has its beginning in the 70's when Albert Bandura - a Canadian psychologist known by many as the originator of the theory - defined self-efficacy as "…people's judgments of their capabilities to organize and execute courses of action required to attain designated types of performances" [3].

Bandura argues that self-efficacy beliefs affect:

1. Choice of activities
2. Level of effort expended
3. Persistence in the face of difficulties
4. Performance [2,3]

Self-efficacy beliefs are also malleable and affect a person's intellectual performance. For these reasons, studies concerning self-efficacy are highly relevant and the theory is widely applied in education as well as other activities where skill development is processed [6]. Wiedenbeck [7] among others present a view on self-efficacy as factor of success. When comparing students at the same level of cognitive skill development, a student with higher self-efficacy beliefs is more likely to take on more advanced and progressive challenges and will strive harder to reach goals. With low self-efficacy beliefs an individual tends to exaggerate problems and may therefore undergo stress and depression which can make a solvable problem impossible [1,7].

The theory of self-efficacy gives four sources of information from which a person's self-efficacy beliefs arise:

1. Personal mastery experiences, in which an individual practices a skill
2. Vicarious experience, for example, modeling of a behavior of others
3. Verbal persuasion, such as suggestions that the individual can cope with a situation
4. Emotional arousal, in which one monitors one's level of stress, anxiety, and fear to assess self-efficacy. [2,3,4]

Introductory programming can be taught in different ways to different students – even within the same university. At Uppsala University, the traditional approach is a Java-centric introduction to the topic of imperative programming where the students have lectures, a few programming assignments and a written exam at the end. One of the study programs observed that their students did not perform as expected in this and following courses, which initiated a process to rework the course structure to achieve a course with a constructive alignment between course goals, syllabus and assessment.

The result was a course structure with programming assignments related to the course goals and no written exam at the end as presented in [5]. Instead, students have roughly one assignment per course week to solve, mostly in groups of two, using pair-programming techniques. Assessment is achieved through continuous feedback, weekly sessions, demonstrations and explanation of the assignments. Other differences in the course are less focus on programming language and more about the ability to correctly debug a program and diagnose errors. Students are exposed to three different languages throughout the course to train the ability to bring previous knowledge into a new programming language. Following this course, there are several courses that include elements of programming where the students are mixed with students that have taken other introductory programming courses. Concerns have been raised about that students who have taken the new course are not as well prepared as other students and thus will not perform as well. To study this closer, an important issue is the one of self-efficacy – how well prepared students consider themselves to be when faced with programming tasks.

Research Question

In this report, we study how the self-efficacy among students who have taken the new introductory programming course changes during a year. The research question asked is: *How does the self-efficacy in programming change from the introductory course until one year later?*

Related Literature

In [6], Ramalingam and Wiedenbeck present a 32-item Self-Efficacy Scale for computer programming. This scale as well as the rest of the article is frequently referred to in studies concerning self-efficacy, e.g., [1, 7, 8].

In [6] the Self-Efficacy Scale was handed out to 421 students in the beginning of a C++ programming course as a pre-test. The students were asked to rate their confidence in doing programming task using a Likert scale from 1 (not at all confident) to 7 (absolutely confident). In the end of the course the same scale was administrated to the same student group as a post-test. Unlike Ramalingam and Wiedenbeck we administrated the pre-test and post-test to two different groups: 2nd year STS students (pre-test) and 3rd year STS students.

Ramalingam and Wiedenbeck [6] assessed their scale and found it to be highly reliable, with a score of 0.98. Looking at their results an increase in self-efficacy between the pre-test and post-test is found, especially among the students with initial low self-efficacy. No substantial difference was found between males and females.

Since Ramalingam and Wiedenbeck [6] studied C++ Programming students, a closer look at the study made by Askar and Davenport [1], including 326 Java Programming Computer Engineering students, gave us wider perspective on how to conduct our study. In [1], Askar and Davenport developed an instrument assessing Java Programming Self-Efficacy from the self-efficacy scale of [6]. Like the test mentioned above this test consisted of 32 items and the reliability was even greater (0.99). The results show with a significant difference that the males' self-efficacy was higher than the females'. The overall self-efficacy score increased with the students' experience, frequency of computer usage, as well as mother's and siblings' computer usage.

Data collection

Data was collected via an online survey that was e-mailed to students in both students groups, STS-2 and STS-3. There was a total of 40 questions in the survey. In five of the questions, the students are asked to answer (on a scale 1 to 4) how well they think they meet five specific course goals of the introductory programming course:

1. Analyze, and design solutions for simple problems
2. Formulate a strategy for managing larger problems
3. Systematic debugging
4. Read, understand and modify small parts of large amounts of code written by others
5. Explain the general principles for how a computer is structured and operates

Three of the questions asked for gender, programming experience and what student group they belonged to. 32 of the questions in the survey have the purpose of evaluating the students' self-efficacy using the exact same questions used in [1], where students grade their confidence on a scale 1 (not at all confident) to 7 (absolutely confident). These questions are developed from the self-efficacy test presented in [6]. Self-efficacy tests regarding programming are often based on the questions in [6]; by using questions from that test, the survey in this study becomes easier to compare with results from other studies.

Participants

The students are engineering students within the sociotechnical systems engineering (STS) program at Uppsala University. Before the introductory programming course, they have done basic programming tasks in two previous courses, though without learning how to program. Their introductory programming course are placed in their third semester and it is a semester-long course with 11 assignments where they first learn Python and later in the course move to Java. There is also one assignment in which they are exposed to assembler programming for a MIPS architecture. During the year following the introductory programming course, they encounter programming tasks in several courses, e.g., Scientific computing II and Object oriented design. These courses are taught jointly with students from other engineering programs. The student groups in the second and third year of the STS program that were asked to participate in this study are described as follows:

	Men	**Women**	**Total**
STS-2	33	20	53
STS-3	14	10	24
Total	47	30	77

Table 1: Participants data

Due to the time limits, it has not been possible to follow a group of students throughout a year. Instead, we have collected answers from two student groups – one that currently takes the introductory programming course (STS-2) and one that took the course one year ago (STS-3). The course teacher and course structure have remained the same as previous year. The main difference between the two student groups are that one of them have since their introductory programming course also taken other courses where they had to solve programming task.

Statistical analysis

The answers from the self-efficacy questions were analyzed using a t-test to examine if there exists a statistical significant difference in self-efficacy between the students in STS-2 and the students in STS-3. The self-efficacy questions were analyzed both by taking all questions into consideration, but also by grouping the questions into different factors. These factors were the same as in [6]:

- Factor 1: Independence and persistence
- Factor 2: Complex programming tasks
- Factor 3: Self-regulation
- Factor 4: Simple programming tasks

The answers from the questions about course goals were also analyzed using a t-test to determine if there exists a significant difference between the two groups. The self-efficacy questions and the questions about the course goals were analyzed completely separately. P-values less than 0.05 are considered to indicate a statistically significant difference.

Results

Cronbach's alpha constant for the course goal scores was 0.79. The item-total correlation calculated through Pearson varied between 0.47 and 0.66 for the course goals. Cronbach's alpha constant for the self-efficacy scores was 0.97. The item-total correlation for the self-efficacy varied between 0.43 and 0.87.

Course goal	**STS-2 Mean (N=33)**	**STS-2 SD**	**STS-3 Mean (N=13)**	**STS-3 SD**	**Difference Mean**	**t-value**	**p-value**
Goal 1	3.21	0.59	3.69	0.46	+0.48	3.936	<0.001
Goal 2	2.82	0.80	3.08	0.83	+0.26	1.433	0.159
Goal 3	2.73	0.75	3.08	0.47	+0.35	2.370	0.022
Goal 4	2.79	0.73	2.77	0.70	-0.02	0.116	0.908
Goal 5	2.06	0.78	2.46	0.84	+0.40	2.238	0.030
All goals	2.72	0.82	3.02	0.79	+0.29	2.390	0.018

Table 2. Means, standard deviations, and difference for course goal scores.

There exists a significant difference between the student groups for goals 1, 3 and 5. For the course goals overall there exist a significant difference between the groups where STS-3 have a higher score than STS-2.

Factor	**STS-2 Mean (N=33)**	**STS-2 SD**	**STS-3 Mean (N=13)**	**STS-3 SD**	**Difference Mean**	**t-value**	**p-value**
Factor 1	4.30	1.71	4.16	1.54	-0.14	0.591	0.555
Factor 2	3.79	1.78	3.80	1.63	+0.01	0.058	0.954
Factor 3	3.89	1.65	4.73	1.53	+0.84	3.356	<0.001
Factor 4	4.98	1.78	4.80	1.74	-0.18	0.702	0.483
All questions	4.27	1.81	4.29	1.68	+0.03	0.113	0.910

Table 3: Means, standard deviations, and difference for self-efficacy groups.

There exists a significant difference between the student groups for category 3. For all the overall self-efficacy score there exist no significant difference between STS-3 and STS-2. Means, standard deviations and differences for the individual questions about self-efficacy are presented in table 4. The questions used for self-efficacy scoring can be found in the appendix of [1].

Question	STS-2 Mean (N=33)	STS-2 SD	STS-3 Mean (N=13)	STS-3 SD	Difference Mean	t-value	p-value
Q1	4.36	1.79	4.23	1.48	-0.13	0.354	0.725
Q2	4.73	1.52	4.69	1.07	-0.03	0.114	0.910
Q3	4.48	1.60	4.77	1.25	+0.28	0.862	0.393
Q4	6.03	1.49	6.00	1.57	-0.03	0.089	0.929
Q5	6.18	1.31	5.77	1.80	-0.41	1.218	0.230
Q6	5.67	1.57	5.62	1.78	-0.05	0.139	0.890
Q7	4.36	1.75	4.92	1.27	+0.56	1.574	0.123
Q8	3.00	1.63	3.08	1.82	+0.08	0.202	0.841
Q9	5.06	1.77	3.69	1.94	-1.37	3.334	0.002
Q10	3.94	1.65	3.54	1.34	-0.4	1.165	0.250
Q11	3.21	1.53	2.77	1.67	-0.44	1.250	0.218
Q12	3.39	1.74	3.00	1.18	-0.39	1.135	0.263
Q13	3.97	1.70	4.00	1.47	+0.03	0.084	0.933
Q14	4.06	1.74	3.46	1.50	-0.6	1.621	0.112
Q15	4.85	1.67	4.62	1.44	-0.23	0.656	0.515
Q16	5.00	1.48	4.69	1.43	-0.31	0.942	0.351
Q17	3.64	1.74	4.08	1.38	+0.44	1.221	0.229
Q18	3.12	1.63	3.46	1.22	+0.34	1.024	0.311
Q19	4.79	1.47	3.77	1.37	-1.02	3.179	0.003
Q20	3.15	1.44	3.31	1.14	+0.16	0.524	0.603
Q21	5.09	1.33	4.38	1.60	-0.71	2.199	0.033
Q22	4.88	1.30	4.77	1.53	-0.11	0.353	0.726
Q23	5.45	1.37	5.08	1.49	-0.38	1.192	0.240
Q24	3.48	1.56	3.31	1.59	-0.18	0.505	0.616
Q25	3.91	1.83	4.62	1.15	+0.71	1.964	0.056
Q26	3.42	1.71	4.31	1.32	+0.88	2.512	0.016
Q27	3.45	1.63	4.31	1.20	+0.85	2.567	0.014
Q28	3.42	1.65	3.46	1.15	+0.04	0.112	0.911
Q29	3.64	1.82	4.46	1.69	+0.83	2.080	0.043
Q30	4.36	1.61	4.69	1.86	+0.33	0.862	0.393
Q31	4.33	1.39	5.62	1.27	+1.28	4.267	<0.001
Q32	4.03	1.60	4.85	1.41	+0.82	2.381	0.022

Table 4: Means, standard deviations, and difference for self-efficacy questions.

There exist a significant difference between STS-2 and STS-3 for questions 9, 19, 21, 26, 27, 29, 31 and 32. For questions 9, 19 and 21, the STS-2 group have a higher score while for questions 26, 27, 29, 31 and 32, the STS-3 group have a higher score.

Conclusion and Discussion

Self-Efficacy

The self-efficacy metric is based on a 32-question form where the questions were grouped into 4 different categories to indicate different types of skills. It is thus possible to discuss self-efficacy in specific skills as indicated by a single question, a group of questions, or as a whole by looking at the mean over all 32 questions.

Approaching this in a top-down manner, we start by looking at self-efficacy as a whole, i.e., by looking at the mean score over all 32 questions for both student groups. This comparison shows no statistically significant difference – neither positive, nor negative – between the two student groups. The overall self-efficacy in basic programming skills seems to remain the same one year after the introductory programming course (mean 4.27 and 4.29, respectively).

A closer look at the different groups of questions shows a statistically significant increase in self-efficacy in category 3 (Self-regulation), while the other categories (Independence, Complex tasks, Simple tasks) show no significant change. It should be noted though, that the self-efficacy in group 1 and 4 (Independence and Simple tasks) were already high in the STS-2 group of students (a mean of 4.3 and 4.8 respectively).

Bringing the discussion down to individual questions, there are a few specific skills where we observe a significant increase in self-efficacy between the two student groups:

- Q26: *I could come up with a suitable strategy for a given programming project in a short time*
- Q27: *I could manage my time efficiently if I had a pressing deadline on a programming project*
- Q29: *I could rewrite lengthy and confusing portions of code to be more readable and clear*
- Q31: *I could find ways of motivating myself to program, even if the problem area was of no interest to me*
- Q32: *I could write a program that someone else could comprehend and add features to at a later date*

These questions are mainly about larger programming projects, which students in their second year had not yet encountered when answering the questions.

There are also a few specific skills where we can observe a significant decrease in self-efficacy between the two groups:

- Q9: *I could write a small Java program given a small problem that is familiar to me*
- Q19: *I could complete a programming project if someone showed me how to solve the problem first*
- Q21: *I could complete a programming project if I could call someone for help if I got stuck*

Apparently, students are not as convinced about their ability to complete a programming project one year after their introductory course as they were before. A speculation is that their increase in experience from larger programming project also have taught them about how hard they can be to complete even if you know how.

Course goals

By observing changes in how students perceive their own capabilities with respect to the course goals, we get an indication about the persistence of skills learned in the introductory course.
Looking at the goals as a whole, STS-3 show a statistically significant increase in the skills expressed in the course goals. Bringing it down to the actual skills, there are three skills in particular where students report a significant increase during the year after the introductory course:

- Analyze, and design solutions for simple problems
- Systematic debugging
- Explain the general principles for how a computer is structured and operates

Although there was a big change in the ability to understand how a computer operates, this course goal is by far the one that students report as their weakest ability.

Overall impressions

We have observed no statistically significant negative changes neither in the students' self-efficacy, nor in the skills reflected by the course goals. There is a significant increase in the belief to meet the course goals between the student groups. Mean values in answers received are all in the upper half of the grading scale (3rd quartile for self-efficacy, 3rd and 4th quartile for course goals).

Sources of error

We have roughly 60% of the students participating in the study, which means that it is hard to draw general conclusions that include all students. As the answers were anonymous, the correlation between who have participated and their performance in courses is unknown to us. A difference between the two student groups is that the second year students take introductory programming as a mandatory course while the third year students have chosen a track within the STS program that includes programming to a larger extent than other tracks.

Future work

Even if this study is limited by the time limits under which it was carried out, there are preliminary results that show a strong sense of self-efficacy among STS students and no significant degradation in the year that follows the introductory course. For future studies, it would be interesting to do a longitudinal study where you follow the same student group rather than asking students from two different years about their self-efficacy beliefs. It would also be interesting to compare these results with students from other study programs where programming is not a central topic, and to relate self-efficacy to course performances.

References

1. P. Askar, D. Davenport, *An Investigation of Factors Related to Self-Efficacy for Java Programming Among Engineering Students*, The Turkish Online Journal of Educational Technology – TOJET January 2009 ISSN: 1303-6521 Vol 8 Issue 1 Article 3.
2. A. Bandura, *Self-Efficacy: Toward a Unifying Theory of Behavioral Change*, Psychological Review, 84:2 pp. 191-215, 1977.
3. A. Bandura, *Social Foundations of Thought and Action*, Prentice Hall, Englewood Cliffs, New Jersey, 1986.
4. M.E. Gist and T.R. Mitchell, Self-Efficacy: A Theoretical Analysis of Its Determinants and Malleability, *Academy of Management Review*, 17, pp. 183-211, 1992.
5. A. Pears, *Conveying Conceptions of Quality through Instruction*, In 7th International Conference on the Quality of Information and Communications Technology, 2010.
6. V. Ramalingam, S. Wiedenbeck, *Development and Validation of Scores on a Computer Programming Self-efficacy Scale and Group Analyses of Novice Programmer Self-efficacy*,Educational Computing Research, vol 19(4) 367-381, 1998.
7. S. Wiedenbeck, *Factors Affecting the Success of Non-Majors in Learning to Program,* In Proceedings of the 2005 international Workshop on Computing Education Research, ICER, pp. 13-24. 2005
8. S. Wiedenbeck, D. LaBelle, and V. Kain, *Factors Affecting Course Outcomes in Introductory Programming*, In 16th Annual Workshop of the Psychology of Programming Interest Group (PPIG16), pp. 97-109, 2004
9. B.J.Zimmerman, *Self-Efficacy and educational Development, in Self-Efficacy in Changing Societies*, A. Bandura (ed.) Cambridge University Press, Cambridge, pp. 203-231, 1995.