An Instrument to Assess Self-Efficacy in Introductory Algorithms Courses

Holger Danielsiek
Department of Computer Science
Westfälische Wilhelms-Universität
Münster, Einsteinstr. 62,
48149 Münster, Germany
holger.danielsiek@uni-muenster.de

Laura Toma
Department of Computer Science
Bowdoin College
8650 College Station
Brunswick, ME 04011, USA
ltoma@bowdoin.edu

Jan Vahrenhold
Department of Computer Science
Westfälische Wilhelms-Universität
Münster, Einsteinstr. 62,
48149 Münster, Germany
jan.vahrenhold@uni-muenster.de

ABSTRACT

We report on the development and validation of an instrument to assess self-efficacy in an introductory algorithms course. The instrument was designed based upon previous work by Ramalingam and Wiedenbeck and evaluated in a multi-institutional setup. We performed statistical evaluations of the scores obtained using this instrument and compared our findings with validated psychometric measures. These analyses show our findings to be consistent with self-efficacy theory and thus suggest construct validity.

CCS CONCEPTS

• **Social and professional topics → Computer science education**; *Student assessment*;

KEYWORDS

Computer Science Education; Self-Efficacy; Algorithms

1 INTRODUCTION

Self-efficacy, introduced by Bandura [2], is a self-belief concept to which, among other effects, considerable influences on academic achievement have been attributed—see, e.g., Pajares and Schunk [31] and the references therein. According to Bandura, self-efficacy, i.e., the expectation of personal efficacy, determines "whether coping behavior will be initiated, how much effort will be expended, and how long it will be sustained in the fact of obstacles and aversive experiences" [2, p. 191]. In the light of recent reports regarding mental problems and decreasing resilience among college students [19, 40], a positive correlation between self-efficacy and coping behaviour makes investigating self-efficacy in undergraduate computing classes worthwhile not only from a purely achievement-oriented point-of-view.

ICER '17, August 18-20, 2017, Tacoma, WA, USA

ACM ISBN 978-1-4503-4968-0/17/08…$15.00
https://doi.org/10.1145/3105726.3106171

It has been argued that self-efficacy can be influenced by corresponding teaching methods—see, e.g., [1, 35, 42] and the references therein. For the development and assessment of such methods, validated instruments for measuring self-efficacy are needed. While Bandura describes the concept of self-efficacy in a domain-unspecific way, it is apparent that coping strategies and techniques for domain-specific challenges should be assessed with domain-specific instruments. It could be argued that developing a course-specific self-efficacy instrument for every single course in the curriculum might not be needed. Given, however, that AP courses or computer science classes in higher secondary education usually focus on programming and programming-related constructs, an algorithms course is likely to pose challenges even to those with prior experience in computer science. Hence, even students who entered college well-prepared may need to overcome obstacles and experience averse situations. For understanding how to cope with this, an instrument specific to to course it is applied in is mandatory.

Even though self-efficacy influences academic achievement, efficacy expectations alone are not sufficient [2, 25]. In addition to efficacy expectations, matching outcome expectations are needed. An outcome expectation is concerned with believing that a certain behaviour will result in a certain outcome, whereas an efficacy expectation is concerned with believing that one can execute a certain behaviour (that potentially leads to a certain outcome) [2]. Put differently, outcome expectations are needed to align one's behaviors with a certain goal, but it is a different issue to be convinced that one can actually do what is needed to reach these goals. In our study, we focused on efficacy expectations.

Self-efficacy does not necessarily manifest itself in "heightened grades" but can be seen as "an important measure of what students 'get'" [42, p. 374] from a class. Consequently, measuring self-efficacy is different from measuring academic performance.

Our research question was how to develop and conceptually validate an instrument to assess self-efficacy in the context of an algorithms course. We discuss the design of such an instrument, an exploratory factor analysis along with an investigation of the internal consistency, and—through comparing our findings with self-efficacy theory and other measures—its construct validity. As the validation of such an instrument usually takes multiple iterations, the research questions of identifying factors that influence the measured construct and of developing corresponding interventions could not be addressed during the first stages of the development reported upon here and thus are subject of future work.

2 RELATED WORK

The concept of self-efficacy has received a considerable amount of attention in educational psychology (see, e.g., [31] and the references therein): according to Google Scholar, Bandura's seminal paper has been cited over 43,000 times. In contrast, the systematic study of self-efficacy has received comparatively little attention in computer science education with an exclusive focus on introductory programming self-efficacy. Ramalingam and Wiedenbeck [36] developed a 32-item instrument for measuring computer programming self-efficacy. This instrument has subscales for four empirically derived factors, interpreted as "Independence and Persistence", "Simple Programming Tasks", "Self-Regulation", and "Complex Programming Tasks" [36]. Using this instrument, Zingaro [42] showed that peer instruction positively contributes to self-efficacy in a CS1 course. Ramalingam et al. [35] built upon their previous work to link previous experiences, self-efficacy, and mental models as factors influencing the performance in an introductory programming course. Another self-efficacy scale was developed by Quade [34], this scale, however, is not specific to course contents and instead includes more general academic and career expectations. In a small-scale, qualitative study, Kinnunen and Simon [24] investigated the self-efficacy of computer science freshmen using a grounded-theory approach; they focussed on the role of programming assignments.

The *growth mindset*-theory [14] has been the subject of a number of recent papers in the computing education research community, e.g., [28, 38]. This theory is concerned with whether or not a learner believes that intelligence and abilities can be developed. This theory is related to self-efficacy, there are, however, important differences: For example, a learner may have a fixed mindset, i.e., believe that abilities and inteligence are inherited and cannot be changed, but still report a high self-efficacy for certain tasks. On the other hand, a growth mindset supports the development of self-efficacy and vice versa—see the literature review by Farrington et al. [15].

3 PARTICIPANTS AND ADMINISTRATION

When composing the study group for developing an instrument to assess the self-efficacy in algorithms, we had to balance different aspects: on one hand, a large enough population should be considered to work in similar conditions as Ramalingam and Wiedenbeck who used $n = 421$ responses for the design of their scales. On the other hand, noting that Ramalingam and Wiedenbeck aggregated "eight sections [...] taught by seven different instructors" [36, p. 369] we also aimed at using courses taught by different instructors to reduce possible bias. Taking into account both enrollment measures and mission goals (the two of the four criteria suggested by Rawson et al. [37] relevant for our context), we ended up with four different participating institutions. Three of these (Bowdoin College, Washington & Lee University, and Williams College) are located in the United States, the other institution (Westfälische Wilhelms-Universität Münster) is located in Europe. The three US schools have been rated consistently among the top undergraduate institutions, the computer science department at Westfälische Wilhelms-Universität Münster has a comparable department size and similar enrollment measures and student-to-faculty ratio as the other institutions. All instructors were tenured research faculty; the second author taught the course at Bowdoin College.

The fact that this set of participants is not representative of the general student population was a deliberate design choice. Due to the relative homogeneity of the student population, more external factors could be ruled out for examining the construct validity of the instrument. Moreover, self-efficacy theory predicts that gains will be stronger for those with low initial self-efficacy [2]. Hence, an instrument that is developed based upon data from participants assumed to show a relatively high initial self-efficacy[1] and that then can be shown to be sensitive to even small changes, is of additional value. When interpreting the results of future applications of the instrument, however, the characteristics of the respective study group should be kept in mind.

The size of the student population from which fully completed responses were obtained during the pre-course study that led to the construction of the subscales for our instrument was $n_{\text{pre-course}} = 362$. Of these, 53 participants were from US institutions while 309 participants were from Westfälische Wilhelms-Universität Münster. The large number of students from Westfälische Wilhelms-Universität Münster is explained by the fact that the algorithms course is a second-semester course attended by a variety of non-majors, including students with Mathematics, Physics, or Business Information Management majors; all students had taken the same "Introduction to Programming" course in the preceding term and, hence, were indistinguishable from the point of previous exposure to computer science concepts in college. The US students had taken "Introduction to Computer Science" and "Data Structures" courses prior to the "Algorithms" courses considered. For the validity of the study, we note that with the exception of dynamic programming (see Section 6) all students were exposed to the same concepts, algorithms, and competence requirements.

The instrument was administered in-class and in a paper-based form as the experiences from a small pilot study as well as from other surveys led to expect a high response rate. On each sheet of paper, the students were asked to give a one-way hashcode based upon information privy to the students only; this code was used to match the otherwise anonymous responses from the pre-course survey and the post-course survey. The pre-course surveys were administered during the first week of classes, the post-course surveys were administered during the last week of classes.

As participation was voluntary and, more importantly, attendance in class was not required, we had $n_{\text{post-course}} = 130$ responses to the post-course questionnaires. The number of matched answers from all pre- and post-course questionnaires was $n_{\text{matched}} = 107$ (28 US/79 European). Preempting the analyses detailed in Section 6 where we will present the pre- and post-course scores in aggregated form, i.e., for all participants, and detailed by continent, we note that the nominal imbalance of the study population did not affect the (significance) of the results reported.

4 DESIGN OF THE INSTRUMENT

In the design of our instrument, we started from the scale provided by Ramalingam and Wiedenbeck [36]. We examined each item and checked whether the constructs behind it could be transferred to an algorithms course. This led to the exclusion of almost half of the items that referred, e.g., to the usage of reserved words in a

[1] Table 4 indeed shows rather high initial scores.

programming language, the object-oriented paradigm, or using third-party functions or libraries. We also decided not to restate the "self-regulation" subscale in the context of algorithms as, after Ramalingam and Wiedenbeck's paper, self-determination theory has led to validated psychometric instrument for measuring self-regulation [4]; also, the concept of a "deadline for a programming project" [36] has no direct counterpart in an algorithms course.

Depending on the school and the preferences of instructors, the main focus of an algorithms course may vary between the details of the implementation in a particular programming language to formal proofs of correctness. Furthermore, it is not uncommon in Europe to teach both data structures and (non-advanced) algorithms in one course and defer more advanced topics, such as dynamic programming or randomization to upper-level courses. Each instructor may have personal preferences for which algorithms to explain in more detail. Finally, the assignment of algorithms to a particular introductory course is not fully agreed upon even within the scope of the ACM computing curriculum [21].

To make our instrument as broadly applicable as possible, we thus decided to not focus on particular algorithms, e.g., Quicksort or the Floyd-Warshall algorithm. Doing so, we also tried to avoid the pitfall of soliciting a self-reported assessment of very specific skills such as executing a particular algorithm as opposed to the intended self-reported self-efficacy related to a more general understanding of topics in algorithms. Based upon curricular aspects and previous work on topics in algorithms prone to misconceptions [17, 39], we constructed items pertaining to runtime analysis, divide-and-conquer, and dynamic programming; see Table 1.

We deliberately decided to only include items that could be operationalized easily. In particular, even though part of the motivation to perform well in an algorithms course is that major companies include "problem solving" items in their assessments, we refrained from including more generic "problem solving" or "abstraction" items into our instrument.[2] A very practical reason for this is that the instrument, just like Ramalingam and Wiedenbeck's instrument, was designed to be administered at a very basic college level or maybe even advanced placement level and thus should be refined to very basic aspects. More importantly, though, recent research on abstraction in computer science classes shows "a strong tendency, by many senior students, to remain on low levels of abstraction, even after realizing abstraction in a variety of CS courses" [18, p. 242] and identifies a lack of practicing abstraction in algorithms courses. Also, the development of abstraction strategies appears to take a longer time than the course of a semester [32].

The existence of a "problem solving" scale in Mathematics seems to contradict the above reasoning. Indeed, Pajares and Miller [30] report having used Dowling's *Mathematics Confidence Scale* [12]; another set of items has been proposed by Dunham [13]. A closer look at these items, however, reveals that the term "problem solving" is used differently: the items in Dunham's scale present a mathematical problem of the form "Solve $\frac{2}{x+3} - 2 > 0$" [13, p. 123] and ask the subject about the confidence of correctly selecting one of five given answers. The items in Dowling's scale[3] are of the form "A living room set consisting of one sofa and one chair is priced at $200. If the price of the sofa is 50% more than the price of the chair, find the price of the sofa" [20, Table 3]. In contrast, the task of "solving a problem" in an algorithms course usually includes modeling the problem, devising an algorithm (or another mathematical construct), proving its correctness, and then applying it to the given setting, see, e.g., [18]. Hence, our above reasoning stands.

As discussed above, Ramalingam and Wiedenbeck had identified "Independence and Persistence" as the factor with the highest Eigenvector. The corresponding items concerned "completing a programming project" with various levels of support. We addressed this by constructing items for both algorithm design and runtime analysis that assumed matching levels of available support. A small-scale pilot study at Bowdoin College (Fall 2015) showed that administering the instrument did not pose organizational obstacles. Also, no student reported difficulties regarding understanding the items.

5 EXPLORATORY FACTOR ANALYSIS

To assess the construct validity of our instrument and to be able to better interpret the results, we followed Ramalingam and Wiedenbeck and performed an exploratory factor analysis of the pre-course data.[4] In line with the previous approach, data was extracted using principal axis factoring and, as the factors could not be expected to be independent, a direct oblimin factor rotation.

With 21 items and $n_{\text{pre-course}} = 362$ responses to the pre-course questionnaire, the overall Kaiser-Meyer-Olkin (KMO) measure was 0.908 with individual KMO measures all greater than 0.846, classifications of "meritious" to "marvellous" [22]. Bartlett's Test of Sphericity was statistically significant ($p < 0.001$, $\chi^2 = 6027.217$), indicating that the data was likely factorizable. The analysis revealed four factors with Eigenvalues greater than 1 which explained 44.3%, 9.3%, 7.9%, and 4.7% of the total variance, respectively. Together, these factors explained 66.3% of the total variance. These factors met the interpretability criterion and were thus retained.

The interpretation of the data was consistent with what the instrument was designed to measure indicating construct validity. The data showed strong loading of items referring to designing an algorithm with varying degrees of support on Factor 1. This "Algorithm Design" factor accounted for 44.3% of the variance and included an item referring to finding counterexamples for an algorithm known to be incorrect; this item, however, has a low loading factor of 0.390. Five items referring to understanding and applying the divide-and-conquer and the dynamic programming paradigm strongly loaded on Factor 2. This factor accounted for 9.3% of the variance and is referred to as "Advanced Paradigms". Four items referring to analyzing the runtime of an algorithm with varying degrees of support strongly loaded on Factor 3. This factor accounted for 7.9% of the variance and is referred to as "Runtime Analysis". The remaining six items, referring to producing and tracing pseudocode, strongly loaded on Factor 4. This factor accounted for 4.7% of the variance and is referred to as "Pseudocode Writing and Tracing".

The details, given in Table 1, also show that there are two items with low primary loadings: the primary loading of Item 10 "Mentally trace through the execution of an iterative algorithm given to me" (Factor "Pseudocode Writing and Tracing") is only marginally

[2] This is in contrast to a deliberately generic instrument to assess self-efficacy in Engineering Design [7].

[3] We report on Hackett and Betz' adaptation [20].

[4] All statistical analyses reported upon were performed using IBM SPSS™ 24 using 95% confidence intervals.

No.	Item Description	I	II	III	IV	h^2	*M*	*SD*
Factor 1: Algorithm Design ($\alpha = 0.898$)								
6	Come up with an algorithm if I could call someone for help if I got stuck.	**.916**	-.044	.051	.004	.757	5.27	1.310
7	Come up with an algorithm once someone helped me get started.	**.856**	-.043	-.043	-.007	.734	5.17	1.311
8	Come up with an algorithm if I had a lot of time.	**.830**	-.057	-.094	-.035	.704	5.35	1.335
4	Organize and design an algorithm in a modular manner.	**.624**	.210	.068	.031	.533	*4.15*	1.505
9	Find ways of overcoming the problem if I got stuck while coming up with an algorithm.	**.444**	.087	-.222	.182	.591	4.42	1.311
15	Come up with a counterexample for an algorithm known to be incorrect.	**.390**	.158	-.144	.169	.483	4.20	1.427
Factor 2: Advanced Paradigms ($\alpha = 0.881$)								
18	Understand the dynamic programming paradigm.	-.077	**.943**	-.032	-.025	.834	3.43	1.725
19	Comprehend a dynamic programming algorithm.	-.023	**.942**	.040	-.041	.818	3.43	1.673
20	Write down a recursive definition of the optimal solution for a dynamic program.	.042	**.799**	-.009	-.070	.645	*3.02*	1.536
17	Understand the divide-and-conquer paradigm.	.091	**.506**	-.096	.196	.505	4.99	1.541
5	Comprehend a complex divide-and-conquer algorithm.	.303	**.362**	-.064	.191	.532	3.90	1.720
Factor 3: Runtime Analysis ($\alpha = 0.929$)								
12	Analyze the running time of an algorithm once someone else helped me get started.	.040	-.045	**-.937**	-.058	.843	4.59	1.481
13	Analyze the running time of an algorithm if I had a lot of time to do so.	.005	-.005	**-.914**	-.028	.843	4.59	1.481
11	Analyze the running time of an algorithm if I could call someone for help if I got stuck.	-.063	-.009	**-.912**	-.014	.755	4.56	1.557
14	Find ways of overcoming the problem if I got stuck at a point while analyzing the running time of the algorithm.	.016	.084	**-.746**	.059	.677	*4.11*	1.448
Factor 4: Pseudocode Writing and Tracing ($\alpha = 0.871$)								
1	Write a pseudocode description for computing the average of three numbers.	-.117	-.055	.041	**.984**	.801	5.77	1.432
2	Write a pseudocode description for solving a small problem that is familiar to me.	-.029	-.069	-.005	**.940**	.823	5.68	1.294
3	Write a pseudocode description for solving a reasonably complex problem that is only vaguely familiar to me.	.198	.023	.014	**.665**	.637	4.38	1.410
16	Write a pseudocode description for sorting *n* numbers.	.143	.100	-.045	**.549**	.505	4.99	1.541
21	Write a pseudocode description for binary search in an ordered array of *n* numbers.	.077	.236	-.156	**.364**	.415	*4.06*	1.788
10	Mentally trace through the execution of an iterative algorithm given to me.	.261	.026	-.276	**.307**	.501	5.14	1.418

Table 1: Factor pattern coefficients, communality estimates, means, and standard deviations for the algorithms self-efficacy items ($n_{\text{pre-course}} = 362$). For each item, the highest factor pattern coefficient above 0.300 is shown in bold. For each of the resulting factors, alpha reliability estimates are given in parentheses. The underlined means display the highest value in each factor which represents the most confident item within this factor. The lowest mean is displayed in italics.

	Mean	*SD*	Factor 1	Factor 2	Factor 3	Factor 4
Algorithm Design	4.758	1.114	1.000			
Advanced Paradigms	3.585	1.420	0.559**	1.000		
Runtime Analysis	4.470	1.366	0.593**	0.468**	1.000	
Pseudocode Writing and Tracing	5.004	1.160	0.688**	0.475**	0.529**	1.000

Table 2: Spearman's rank-correlation coefficients for the exploratory factor analysis from the pre-course data ($n_{\text{pre-course}} = 362$; **: correlation significant at $p < 0.01$).

in excess of 0.3. However, as the next highest loading was well below 0.3, we decided not to omit this items and its assignment to the factor. For Item 5 "Comprehend a complex divide-and-conquer algorithm", both the primary loading (0.362; Factor "Advanced Paradigms") and the secondary loading (0.302; Factor "Algorithm Design") were higher than 0.3. However, the secondary loading was only marginally higher that this threshold value, the primary loading was roughly 20% higher than the secondary loading. As, in

Group	n	All Items					Excluding Dynamic Programming Items				
		μ_{pre}	μ_{post}	δ_μ	t	p	μ_{pre}	μ_{post}	δ_μ	t	p
ALL	107	4.56 (1.00)	5.37 (0.95)	0.81 (0.86)	9.714	< 0.0005	4.81 (1.02)	5.55 (0.92)	0.74 (0.85)	8.977	< 0.0005
US	28	4.70 (0.98)	5.97 (0.76)	1.27 (0.87)	7.642	< 0.0005	5.08 (1.05)	5.95 (0.78)	0.87 (0.93)	4.936	< 0.0005
Europe	79	4.50 (1.01)	5.15 (0.91)	0.65 (0.81)	7.180	< 0.0005	4.71 (1.00)	5.41 (0.93)	0.69 (0.82)	7.486	< 0.0005

Table 3: Results of running a paired t-test on the $n = n_{\text{matched}} = 107$ differences between pre- and post-course scores (average over all items). Both mean and standard deviation (in parentheses) are given. Results are reported for the scores derived from all items and the scores derived without the items referring to Dynamic Programming.

addition, the inspection of the items in each group confirmed that the Item 5 corresponded to Item 17 which had a clearly higher loading on the "Advanced Paradigms" Factor than on the "Algorithm Design" factor, we decided to also keep Item 5 in the instrument and assign it to the "Advanced Paradigms" factor.

The reliability of the scores, measured by Cronbach's Alpha, was 0.938 with reliability of the four factors ranging between 0.871 and 0.929. This is slightly less than the reliability of Ramalingam and Wiedenbeck's scores but still high enough to meet clinical standards [5]. The scale showed a high re-test validity in the $n_{\text{post-course}} = 130$ post-course responses: the reliability of all scores was 0.950 with the reliability of the four factors ranging between 0.868 and 0.931.

To possibly account for differences between the institutions, we attempted to separately extract factors based upon the US and European data. However, the data quality for the $n_{\text{post-course,US}} = 32$ US responses was too low (KMO measure: 0.684) to yield reliable results. Thus, we used the above factors derived from all pre-course responses. Table 2 presents the descriptive statistics of the empirically-derived factors and their correlations.

Naturally, the above factors could also be used with different weights that reflect their relative importance for a certain outcome.

6 ANALYSES

We discuss the results of applying the scale and the factors obtained by our exploratory factor analysis. To assess construct validity of our instrument, we relate our findings to self-efficacy theory.

Pre-/Post-Course. According to general self-efficacy theory and previous studies on self-efficacy, an increase in self-efficacy is to be expected over the course of a semester. To verify this, we analyzed the change in the overall, i.e., averaged over all 21 items, pre-course and post-course scores.

According to Fay and Proschan, one should use a t-test "whenever the difference in means is desired for interpretation of the data" [16, p. 19]. Analyzing the data showed two outliers that were more than 1.5 box-lengths from the edge of the box in a boxplot. The first of these values corresponded an increase from 1.5 (pre-course) to 5.85 (post-course). Further inspection revealed that this student had indicated a low confidence ("1") on all items containing technical terms specific to the design and analysis of algorithms. It appears that this student had responded to the cue "If a specific term or task is totally unfamiliar to you, e.g., because it has not been discussed in class, yet, please mark 1" (this cue also had been used by Ramalingam and Wiedenbeck [36, p. 372]), hence, the value was kept in the analysis. The second of these values corresponded to a decrease from 5.0 (pre-course) to 1.16 (post-course). Even though this value was extreme, we could not dismiss the data point as some of the entries were larger than 1 indicating that the student had not simply given an unreflected, default answer.

To assess the influence of the second of these outliers, we ran a t-test with and without this data point included. In both cases, normality was confirmed using a visual inspection of a Normal Q-Q plot.[5] Including the second outlier, the scores increased by 0.812 (95% CI, 0.646 to 0.978) from pre-course to post-course, the increase was statistically significant, $t(106) = 9.714$, $p < 0.0005$. Excluding the second outlier, the scores increased by 0.846 (95% CI, 0.692 to 0.999) from pre-course to post-course, the increase was statistically significant, $t(105) = 10.952$, $p < 0.0005$. We concluded that keeping the extreme outlier did not affect the type or significance of the change and retained it as well.

We then separately investigated the descriptive statistics and the change in scores for the three US institutions[6] and the European institution. For the three US institutions, there were no outliers in the data, as assessed by inspection of a boxplot for values greater than 1.5 box-lengths from the edge of the box. The differences in scores were normally distributed as assessed by visual inspection of a Normal Q-Q plot and the Shapiro-Wilk test ($n = 28$, $p = 0.926$). The scores increased by 1.26 (95% CI, 0.927 to 1.607) from pre-course to post-course, the increase was statistically significant, $t(27) = 7.642$, $p < 0.0005$.

For the European institution, Shapiro-Wilk test failed to confirm normality due to the presence of the extreme outlier discussed above ($n = 79$, $p = 0.004$). Following the above line of reasoning, however, we confirmed normality using visual inspection of a Normal Q-Q plot and kept the data point.[7] The scores increased by 0.651 (95%

[5] The second outlier, however, caused the Shapiro-Wilk test to fail ($n = 107$, $p < 0.001$); deleting this data point would have resulted in the Shapiro-Wilk test confirming normality as well ($n = 106$, $p = 0.165$).

[6] Due to the small sample sizes at each institution and as the courses had been selected to be comparable (see Section 3), we did not perform separate analyses for each institution.

[7] Deleting the extreme data point would have resulted in passing the Shapiro-Wilk test as well ($n = 78$, $p = 0.244$).

		Pre-Course Self-Efficacy				Post-Course Self-Efficacy			
Group	n	Algorithm Design	Advanced Paradigms	Runtime Analysis	Pseudocode Write/Trace	Algorithm Design	Advanced Paradigms	Runtime Analysis	Pseudocode Write/Trace
US female	8	4.85 (1.61)	2.73 (1.56)	5.53 (1.07)	5.60 (0.77)	5.77 (1.00)	6.15 (0.99)	6.03 (0.94)	6.27 (0.62)
US male	20	4.85 (1.22)	3.14 (1.21)	4.90 (1.81)	5.75 (0.72)	5.82 (0.84)	5.89 (0.70)	5.55 (1.19)	6.36 (0.73)
US	28	4.85 (1.31)	3.02 (1.31)	5.08 (1.64)	5.71 (0.72)	5.80 (0.87)	5.96 (1.13)	5.68 (0.69)	6.33 (0.76)
Europe	79	4.84 (1.04)	3.63 (1.45)	4.32 (1.33)	5.01 (1.11)	5.30 (1.05)	4.81 (1.20)	5.13 (1.27)	5.75 (0.94)
ALL	107	4.85 (1.12)	3.48 (1.44)	4.53 (1.45)	5.19 (1.06)	5.43 (1.03)	4.72 (1.32)	5.28 (1.25)	5.91 (0.91)

Table 4: Descriptive statistics for various demographic groups. For each factor, mean and standard deviation (in parentheses) are given. The European course did not cover Dynamic Programming, hence, the post-course scores for the "Advanced Paradigms" self-efficacy factor are considerably lower.

CI, 0.470 to 0.831) from pre-course to post-course, the increase was statistically significant, $t(78) = 7.180$, $p < 0.0005$.

The t-values given indicate that the increase in scores was higher for the US institutions than for the European institutions. However, it is important to note that the curriculum for the European course did not contain dynamic programming as this topic was handled in an advanced algorithms course. Instead, the course covered several data structures topics not contained in the curricula for the US courses. Filtering out the three items related to dynamic programming, the analysis showed the changes to be much more consistent across institutions: the scores for the US courses increased by 0.871 (95% CI, 0.509 to 1.23; $t(27) = 4.936$, $p < 0.0005$) and the scores for the European institution increased by 0.694 (95% CI, 0.510 to 0.879; $t(78) = 7.486$, $p < 0.0005$). Thus, the influence of the two subgroups (US vs. European) did not affect the overall outcome or its significance: Table 3 show that a statistically significant increase could be found for both the whole population and individual groups.

Self-Efficacy Gain for Quartile Groups. Given that only few, if any, students have been exposed to the topics covered in an algorithms course before, it can be assumed that a low initial self-efficacy is not attributed to previous aversive experience. In line with Ramalingam and Wiedenbeck, we thus conjectured that the largest gain in self-efficacy would be observed for those with a low initial self-efficacy.

To investigate this hypothesis, we conducted a one-way ANOVA ($\alpha = 0.05$) to determine whether the difference in pre-course and post-course scores was different for the groups determined by the pre-course scores' quartiles. The group corresponding to the lowest quartile contained 27 participants, the other groups, in increasing order of quartiles, contained 31, 23, and 26, participants, respectively. As discussed above, the data set contained two outliers, one of which could be traced back to apparently erring on the side of caution and checking "not at all confident" for every single item on the pre-course questionnaire, hence, resulting in a extremely large increase in self-efficacy. The other participant showed an extreme drop in self-efficacy, which we have been unable to explain so far. Thus, the data of both participants was retained. Homogeneity of variance, as assessed by Levene's test for equality of variances ($p = 0.076$), was observed. Data is presented as mean ± standard deviation.

The increase in self-efficacy was statistically significantly different for the four groups ($F(3, 103) = 9.974$, $p < 0.001$, partial $\eta^2 = 0.363$). The differences increased from the first (1.46 ± 0.85), to second (0.76 ± 0.98), to third (0.67 ± 0.52), to fourth (0.32 ± 0.58) groups, in that order. Tukey post hoc analysis revealed that the increase from the first to the second (-0.69, 95% CI (-1.22 to -0.16), $p = 0.005$), from the first to the third (-0.77, 95% CI (-1.35 to -0.21), $p = 0.002$), and from the first to the fourth group (-1.13, 95% CI (-1.69 to -0.57), $p < 0.001$) was statistically significant.

Gender. Responding to the literature on gender differences regarding self-assessment in computing [3, 8, 11], we followed up by a one-way ANOVA ($\alpha = 0.05$) for both the overall scores (averaged over all items) as well as for each of the factors. As we only had access to the the gender information of $n_{\text{gender}} = 28$ US participants, the population studied was rather small. Normality for the change in scores was assessed using a Shapiro-Wilk test within each group ($p > 0.05$). There was homogeneity of variances, as assessed by Levene's test for equality of variances ($p = 0.340$). However, no statistically significant differences between the group of $n_{\text{female,US}} = 8$ female participants and the groups of $n_{\text{male,US}} = 20$ male participants ($F(1, 26) = 0.130, p = 0.721$) could be detected; the same holds on the level of the factors. This is consistent with the findings of Ramalingam and Wiedenbeck who could not find a statistically significant difference between (much larger groups of) female and male students w.r.t. computer programming self-efficacy. Their hypothesis that "females entering computer science are members of a self-selected group that tends to have high mathematics [...] experience" [36, p. 379] is supported by Beyer [3] who reported (slightly) higher Math ACT scores for both major and non-major female students in introductory programming courses.

Pre-/Post-Midterm. In each of the US courses considered in this study, students had to take a midterm exam. Since the midterm exam was the first major formal feedback putting self-efficacy in

	Mean	*SD*	Self-Efficacy	Perceived Competence	Interest/Enjoyment	Pressure/Tension
Self-Efficacy	5.761	0.853	1.000			
Perceived Competence	4.147	1.479	0.629**	1.000		
Interest/Enjoyment	4.326	1.132	0.527**	0.662**	1.000	
Pressure/Tension	4.116	1.601	-0.335*	-0.604**	-0.280	1.000

Table 5: Spearman's rank-correlation coefficient for post-midterm self-efficacy and factors from the *Intrinsic Motivation Inventory* ($n_{\text{midterm}} = 46$; *: correlation significant at $p < 0.05$; **: correlation significant at $p < 0.01$).

context with actual performance, we administered our instrument immediately before and after the midterm exam.

The changes as reported by our instrument could not be shown to be statistically significant: 12 of the $n_{\text{midterm,matched}} = 23$ students for which we had complete data saw an increase in self-efficacy while 11 participants saw a decrease. In the light of self-efficacy theory, this is rather unsurprising, since taking this first algorithms exam could either reinforce existing self-efficacy, show performance accomplishments, or give demotivating, negative feedback. Consistent with self-efficacy theory, the gain in self-efficacy from this first calibration point until the end of the course was statistically significant again as assessed by a Wilcoxon signed-rank test[8] ($n = 23$, $n_{\text{incr}} = 16$, $n_{\text{decr}} = 5$, $n_{\text{tie}} = 2$, $z = 2.714$, $p = 0.007$, $r = 0.40$). This can be related to both performance accomplishments later in the course and vicarious experiences as "seeing others perform threatening activities without adverse conditions [...] [can convince observers that they can] achieve at least some improvement" [2, p. 197]. We conjecture that the small class sizes at the participating US institutions favoured closely observing vicarious experiences.

We relate these observations to the findings by Lishinski et al. [26]. In their research on general self-efficacy in an introductory programming course, they observed the influence of performance feedback on the development of self-efficacy. As they used a general self-efficacy scale taken from the *Motivated Strategies for Learning Questionnaire*, this similarity in observations supports convergent construct validity. Lishinski et al. also found female students to exhibit different development patterns than male students: female students were reported to adjust their self-efficacy before the midterm exams while male students started adjusting their self-efficacy after this point. In our analyses using a one-way ANOVA, however, we could not detect statistically significant differences between male and female students for the changes in scores from pre-course to pre-midterm ($F(1, 21) = 0.21$, $p = 0.887$) and post-midterm to post-course ($F(1, 21) = 0.75$, $p = 0.786$). We hypothesize these divergent observations to be related to the different nature of the introductory programming course studied by Lishinski et al. and the introductory algorithms courses studied by us: female students have been reported to have less confidence in their computing ability than male students [3], also it is known that programming-heavy courses and their assignments induce an emotional burden [23]. Combining these two factors with the fact that the midterm was the first feedback that might positively influence self-efficacy can explain the observations of Lishinski et al.. At the same time, these factors and the self-selection also reported by Beyer et al. [3] is suited to explain why no such gender-based difference could be observed in our study of a much more math-oriented algorithms courses.

For assessing motivational aspects that may influence the self-efficacy after the midterm, we administered a 12-item questionnaire together with the post-midterm questionnaire. The subscales in this instrument were taken from the *Intrinsic Motivation Inventory*—see, e.g., [10]—and referred to "Interest/Enjoyment", "Perceived Competence", and "Pressure/Tension". These scores were checked for correlations with the post-midterm self-efficacy scores as measured by our instrument. Not all variables were normally distributed (Shapiro-Wilk test, $p < 0.05$), so Spearman rank-order correlations were computed. Preliminary analyses showed all relations to be monotonic as assessed by scatterplot.

Table 5 shows that both the "Interest/Enjoyment" and the "Perceived Competence" scores positively correlate with the post-midterm self-efficacy ($n_{\text{midterm}} = 46$). Conversely, the "Pressure/Tension" scores negatively correlate with post-midterm self-efficacy. This is in line with Bandura's theory about the correlations between performance accomplishment and self-efficacy on one hand and emotional arousal and self-efficacy on the other hand [2, p. 195ff.].

In addition, we compared these motivational scores with self-reported midterm graded obtained as part of the post-course evaluation. Reporting these grades was optional, hence, not all students chosen to do so ($n_{\text{reported-grades}} = 28$). As already observed by Wilson and Shrock [41], no statistically significant correlation between grades and self-efficacy (as reported immediately after having completed the exam) could be observed (Spearman's rank-order, $r_s(28) = -0.203$, $p = 0.300$). Running a Spearman's rank-order correlation on "Pressure/Tension" and self-reported grades, however, showed a rather strong correlation ($r_s(28) = 0.545$, $p = 0.003$). As the grades were coded inverted, i.e., $A = 1$, $B = 2$, and $C = 3$, this shows that high pressure/tension correlated with a low grade. This is of particular interest, as all midterm exams were take-home exams, i.e., it can be assumed that time pressure played a much weaker role than for in-class exams.

7 CONSTRUCT VALIDITY

The validation of an instrument usually requires multiple iterations during which the instrument may be refined. In the context of this paper, we only report on our preliminary steps to assess construct validity. In the previous section, we have discussed how the changes in pre- and post-scores related to general self-efficacy theory. To further investigate construct validity, we followed Peter [33] who distinguishes four aspects of construct validity: reliability, convergent validity, divergent validity, and nomological validity.

[8] Due to the small number of subjects for which data was available, we decided not to use a t-test.

The reliability, i.e., how much the instrument is correlated with itself was assessed through the alpha reliabilities within the factors (between 0.871 and 0.929, see Table 1) and the statistically significant correlations between the factors (see Table 2). Both measures showed the necessary reliability conditions to be fulfilled.

As the proposed instrument is, to the best of the authors' knowledge, the first instrument directed towards assessing self-efficacy in the domain of introductory algorithms, a multitrait-multimethod matrix [6], the standard method to assess convergent and divergent validity, could not be computed. Instead, we tried to establish nomological and discriminant validity by relating the obtained scores and other psychometric scores obtained together with the algorithms pre-course questionnaire. As in no case all variables were normally distributed (Shapiro-Wilk test, $p < 0.05$), all correlations were computed as Spearman's rank correlations.

Computer Programming Self-Efficacy. This score was determined by administering the instrument developed by Ramalingam and Wiedenbeck [36]. This instrument was administered in all courses together with our instrument during the pre-course tests. Where needed, we restated the items in terms of the programming language used in the preceding programming course. We found a strong positive correlation between programming self-efficacy and algorithms self-efficacy, ($r_s(99) = 0.772$, $p < 0.001$). At first, this may seem to indicate that the two instrument measure the same construct more than nomological expectations would indicate and, hence, that our instrument is not course-specific enough. However, previous research showed a correlation between performance in introductory programming and introductory algorithms courses [9] indicating that also self-efficacy in these areas may be correlated.

Moreover, divergence was found during the factor analysis. In the original paper, Ramalingam and Wiedenbeck had worked with items asking for the confidence to "complete a programming project" given various degrees of support. These items had been found to constitute a factor labeled "Independence and Persistence". In the design of our instrument, we had restated these items both the "come up with an algorithm" context (Items 6–8) and "analyze the running time" context (Items 11–13). Our exploratory factor analysis (using the same methodology as Ramalingam and Wiedenbeck) clearly separated these two instantiations of Ramalingam and Wiedenbecks' "Independence and Persistence" factor. Instead, the analysis grouped them with different other items indicating that at least on subscale levels the instruments were not measuring identical constructs. We leave to future work whether the domain-specifics overshadow the interpretation of Ramalingam and Wiedenbeck or whether their interpretation of this factor should be revisited.

Self-Regulation. As mentioned in the introduction, we had not included self-regulation items in our instrument. Instead, this score was determined using ten items from Black and Deci's validated scale [4]. Six of these items referred to "Autonomous Regulation", the remaining four items referred to (externally) "Controlled Regulation". We compared these scores with the scores for the "Self-Regulation" factor as determined by Ramalingam and Wiedenbeck's instrument (see above). While positive correlations could be found between this factor and both autonomous and controlled regulation as well as the sum of these, none of these were statistically significant. Again, this suggests a discrimination of our instrument against the instrument proposed by Ramalingam and Wiedenbeck. Also, the missing convergence between Black and Deci's instrument and the "Self-Regulation" factor of Ramalingam and Wiedenbeck raises interesting revalidation questions for future work.

Personality Traits. According to O'Connor and Paunonen [29], the "Conscientiousness" and "Openness to Experience" personality traits are positively associated with academic success. As self-efficacy has been associated with academic success as well [30, 31], we thus investigated whether nomological validity, i.e., a "relationship between measures purported to assess different (but conceptually related) constructs" [33, p. 137f.] could be established. Personality traits scores were determined using sixteen "Conscientiousness" items and eight "Openness to Experience" items from the NEO personality inventory [27]. We found positive correlations between "Conscientiousness" and both programming self-efficacy ($r_s(99) = 0.383$, $p < 0.001$) and post-course algorithms self-efficacy ($r_s(99) = 0.322$, $p = 0.001$); correlation to pre-course algorithms self-efficacy was weaker ($r_s(99) = 0.200$, $p < 0.05$). "Openness to Experience" was positively correlated to pre-course ($r_s(99) = 0.272$, $p = 0.006$) and post-course ($r_s(99) = 0.300$, $p = 0.003$) algorithms self-efficacy; correlation to programming self-efficacy was weaker ($r_s(99) = 0.272$, $p = 0.012$). These findings are consistent with the interpretation of "Conscientiousness" in terms of motivation and the weaker role of "Openness to Experience" as discussed by O'Connor and Paunonen [29]; thus suggesting nomological validity.

Summary. Summarizing the analyses reported in this section, we conclude that reliability could be confirmed statistically and supporting evidence for nomological validity (through other psychometric score) and divergent validity (through comparison with programming self-efficacy and general self-regulation) can be reported. Due to the lack of alternative instruments, convergent validity could not be examined beyond comparing our instrument with an instrument to assess programming self-efficacy.

8 CONCLUSIONS

We have developed an instrument geared towards assessing self-efficacy in an introductory algorithms course. An exploratory factor analysis showed four main factors which we interpret as "Algorithm Design", "Advanced Paradigms", "Runtime Analysis" and "Pseudocode Writing and Tracing". The results of statistical analyses on the first application of this instrument suggest construct validity with respect to general self-efficacy theory. Additional support for construct validity comes from correlations with related psychological factors, such as motivation and personality traits.

To strengthen the validity argument and to increase the confidence in making judgements using this instrument in broader student populations, we plan to extend the context of applicability and to apply this instrument at institutions with a broader range of selectivity. Also, we aim at identifying topics and corresponding items for which extreme changes or no changes at all can be observed when assessing self-efficacy; this will enable us to examine the influence of teaching methodologies on algorithms self-efficacy.

Acknowledgments. We thank the participating instructors for their cooperation and B. Köpcke and J. Seep for entering the data. This work was partially supported by the German Federal Ministry of Education and Research (grant 01PB14007A).

REFERENCES

[1] A. R. Artino Jr. Academic self-efficacy: from educational theory to instructional practice. *Perspectives in Medical Education*, 1:76–85, 2012.

[2] A. Bandura. Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, 84(2):191–215, 1977.

[3] S. Beyer, K. Rynes, J. Perrault, K. Hay, and S. Haller. Gender differences in computer science students. In S. Grissom, D. Knox, D. Joyce, and W. Dann, editors, *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, pages 49–53. ACM Press, 2003.

[4] A. E. Black and E. L. Deci. The effects of instructors' autonomy support and students' autonomous motivation on learning organic chemistry: A self-determination theory perspective. *Science Education*, 84(6):740–756, Nov. 2000.

[5] J. M. Bland and D. G. Altman. Statistics notes: Cronbach's alpha. *British Medical Journal*, 314:572, Feb. 1997.

[6] D. T. Campbell and D. W. Fiske. Convergent and discriminant validation by the multitrait-multimethod matrix. *Psychological Bulletin*, 56(2):81–105, Mar. 1959.

[7] A. R. Carberry, H.-S. Lee, and M. W. Ohland. Measuring engineering design self-efficacy. *Journal of Engineering Education*, 99(1):71–79, Jan. 2010.

[8] J. M. Cohoon. Gendered experiences of computing graduate programs. In I. Russell, S. M. Haller, J. D. Dougherty, and S. H. Rodger, editors, *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, pages 546–550. ACM Press, 2007.

[9] H. Danielsiek and J. Vahrenhold. Stay on these roads: Potential factors indicating students' performance in a CS2 course. In C. Alphonce, J. Tims, M. E. Caspersen, and S. Edwards, editors, *Proceedings of the 47th SIGCSE Technical Symposium on Computer Science Education (SICGSE 2016)*, pages 12–17. ACM Press, 2016.

[10] E. L. Deci, H. Eghari, B. C. Patrick, and D. R. Leone. Facilitating internalization: The self-determination theory perspective. *Journal of Personality*, 62(1):119–142, Dec. 1994.

[11] P. Denny, A. Luxton-Reilly, J. Hamer, D. B. Dahlstrom, and H. C. Purchase. Self-predicted and actual performance in an introductory programming course. In R. Ayfer, J. Impagliazzo, and C. Laxer, editors, *Proceedings of the 15th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2010*, pages 118–122, New York City, 2010. ACM.

[12] D. M. Dowling. *The Development of a Mathematics Confidence Scale and Its Application in the Study of Confidence in Women College Students*. PhD thesis, Ohio State University, 1978.

[13] P. H. Dunham. *Mathematical Confidence and Performance in Technology-Enhanced Precalculus: Gender-Related Differences*. PhD thesis, Ohio State University, 1990.

[14] C. S. Dweck. *Self-theories: Their role in motivation, personality, and development*. Psychology Press, Philadelphia, PA, 1999.

[15] C. A. Farrington, M. Roderick, E. Allensworth, J. Nagaoka, T. S. Keyes, D. W. Johnson, and N. O. Beechum. *Teaching adolescents to become learners. The role of noncognitive factors in shaping school performance: A critical literature review*. University of Chicago Consortium on Chicago School Research, Chicago, 2012.

[16] M. P. Fay and M. A. Proschan. Wilcoxon-Mann-Whitney or t-test? on assumptions for hypothesis testing and multiple interpretations of decision rules. *Statistics Surveys*, 4:1–39, 2010.

[17] J. Gal-Ezer and E. Zur. The efficiency of algorithms – misconceptions. *Computers & Education*, 42:215–226, 2004.

[18] D. Ginat and Y. Blau. Multiple levels of abstraction in algorithmic problem solving. In M. E. Caspersen, S. Edwards, T. Barnes, and D. D. Garcia, editors, *Proceedings of the 48th SIGCSE Technical Symposium on Computer Science Education (SICGSE 2017)*, pages 237–242. ACM Press, 2017.

[19] P. Gray. Declining student resilience: A serious problem for colleges. *Psychology Today*, 2015. Blog post (Sep. 22): `https://www.psychologytoday.com/blog/freedom-learn/201509/declining-student-resilience-serious-problem-colleges`.

[20] G. Hackett and N. E. Betz. Mathematics self-efficacy expectations, math performance, and the consideration of math-related majors. Paper presented at the Annual Meeting of the American Educational Research Association, Mar. 1982. 42 pages.

[21] M. Hertz. What do CS1 and CS2 mean?: Investigating differences in the early courses. In T. Cortina and E. Walker, editors, *Proceedings of the 41st SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2010)*, pages 199–203. ACM Press, 2010.

[22] H. F. Kaiser and J. Rice. Little Jiffy, Mark IV. *Educational and Psychological Measurement*, 34(1):111–117, Sept. 1974.

[23] P. Kinnunen and B. Simon. Experiencing programming assignments in CS1: the emotional toll. In M. E. Caspersen, M. J. Clancy, and K. Sanders, editors, *Proceedings of the 6th International Workshop on Computing Education Research (ICER 2010)*, pages 77–86. ACM Press, 2010.

[24] P. Kinnunen and B. Simon. CS majors' self-efficacy perceptions in CS1: Results in light of social cognitive theory. In M. E. Caspersen, A. Clear, and K. Sanders, editors, *Proceedings of the 7th International Workshop on Computing Education Research (ICER 2011)*, pages 19–26. ACM Press, 2011.

[25] R. W. Lent and S. D. Brown. Towards a unifying social cognitive theory of career and academic interest, choice, and performance. *Journal of Vocational Behavior*, 45(1):79–122, 1994.

[26] A. Lishinski, A. Yadav, J. Good, and R. Enbody. Learning to program: Gender differences and interactive effects of students' motivation, goals, and self-efficacy on performance. In J. Sheard, J. Tenenberg, D. Chinn, and B. Dorn, editors, *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER 2016)*, pages 221–220. ACM Press, 2016.

[27] J. L. Maples, L. Guan, N. T. Carter, and J. D. Miller. A test of the international personality item pool representation of the revised NEO personality inventory and development of a 120-item IPIP-based measure of the five-factor model. *Psychological Assessment*, 26(4):1070–1084, Dec. 2014.

[28] L. Murphy and L. Thomas. Dangers of a fixed mindset: Implications of self-theories research for computer science education. In J. Amillo, C. Laxer, E. M. Ruiz, and A. Young, editors, *Proceedings of the 13th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2008*, pages 271–275, New York City, 2008. ACM.

[29] M. C. O'Connor and S. V. Paunonen. Big five personality predictors of post-secondary academic performance. *Personality and Individual Differences*, 43(5):971–990, 2007.

[30] F. Pajares and M. D. Miller. Role of self-efficacy and self-concept beliefs in mathematical problem solving: A path analysis. *Journal of Educational Psychology*, 86(2):193–203, 1994.

[31] F. Pajares and D. H. Schunk. Self-beliefs and school success: Self-efficacy, self-concept, and school achievement. In R. Riding and S. Rayner, editors, *Perception*, pages 239–266. Ablex Publishing, London, 2001.

[32] J. Perrenet and E. Kaasenbrood. Levels of abstraction in students' understanding of the concept of algorithm: the qualitative perspective. In R. Davoli, M. Goldweber, and P. Salomoni, editors, *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2006*, pages 270–274, New York City, 2006. ACM.

[33] J. P. Peter. Construct validity: A review of basic issues and marketing practices. *Journal of Marketing Research*, 18(2):133–145, May 1981.

[34] A. Quade. Development and validation of a computer science self-efficacy scale for CS0 courses and the group analysis of CS0 student self-efficacy. In *Proceedings of the International Conference on Information Technology: Computers and Communications (ITCCí03)*, pages 60–64. IEEE Computer Society, 2003.

[35] V. Ramalingam, D. LaBelle, and S. Wiedenbeck. Self-efficacy and mental models in learning to program. In R. D. Boyle, M. Clark, and A. Kumar, editors, *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2004*, pages 171–175, New York City, 2004. ACM.

[36] V. Ramalingam and S. Wiedenbeck. Development and validation of scores on a computer programming self-efficacy scale and groups analyses of novice programmer self-efficacy. *Journal of Educational Computing Research*, 19(4):367–381, Dec. 1998.

[37] T. M. Rawson, D. P. Hoyt, and D. J. Teeter. Identifying "comparable" insitutions. *Research in Higher Education*, 18(3):299–310, 1983.

[38] B. Simon, B. Hanks, L. Murphy, S. Fitzgerald, R. McCauley, L. Thomas, and C. Zander. Saying isn't necessarily believing: influencing self-theories in computing. In M. E. Caspersen, R. Lister, and M. Clancy, editors, *Proceedings of the Fourth International Computing Education Research Workshop, ICER 2008*, pages 173–184, New York City, 2008. ACM.

[39] J. Vahrenhold and W. Paul. Developing and validating test items for first-year computer science courses. *Computer Science Education*, 24(4):304–333, Dec. 2014.

[40] R. Wilcon. An epidemic of anguish: Overwhelmed by demand for mental-health care, colleges face conflicts in choosing how to respond. *Higher Education Chronicle*, 2015. Aug. 21, `http://chronicle.com/article/An-Epidemic-of-Anguish/232721`.

[41] B. C. Wilson and S. Shrock. Contributing to success in an introductory computer science course: A study of twelve factors. In H. M. Walker, R. A. McCauley, J. L. Gersting, and I. Russell, editors, *Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education*, pages 184–188. ACM Press, 2001.

[42] D. Zingaro. Peer instruction contributes to self-efficacy in CS1. In J. D. Dougherty, K. Nagel, A. Decker, and K. Eiselt, editors, *Proceedings of the 45th SIGCSE Technical Symposium on Computer Science Education (SICGSE 2014)*, pages 373–378. ACM Press, 2014.