A Systematic Review of Approaches for Teaching Introductory Programming and Their Influence on Success

Arto Vihavainen and Jonne Airaksinen
Department of Computer Science
University of Helsinki
Finland
{ avihavai, jonnaira }@cs.helsinki.fi

Christopher Watson
School of Engineering and Computing Sciences
University of Durham
United Kingdom
christopher.watson@dunelm.org.uk

ABSTRACT

Decades of effort has been put into decreasing the high failure rates of introductory programming courses. Whilst numerous studies suggest approaches that provide effective means of teaching programming, to date, no study has attempted to quantitatively compare the impact that different approaches have had on the pass rates of programming courses. In this article, we report the results of a systematic review on articles describing introductory programming teaching approaches, and provide an analysis of the effect that various interventions can have on the pass rates of introductory programming courses. A total of 60 pre-intervention and post-intervention pass rates, describing thirteen different teaching approaches were extracted from relevant articles and analyzed. The results showed that on average, teaching interventions can improve programming pass rates by nearly one third when compared to a traditional lecture and lab based approach.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Computer science education*

General Terms

Human Factors

Keywords

programming education, introductory programming, cs1, teaching interventions, analysis, systematic review

1. INTRODUCTION

The mean worldwide failure rates of CS1 have been suggested to be as high as one third of students failing the course [3]. A recent study showed that despite advances in pedagogy, the worldwide failure rates of CS1 have not improved over time, and that the failure rates are not substantially influenced by aspects of the external teaching context, such as the programming language taught in the course [25].

Despite decades of research, internal factors based upon traditional learning theories have also failed to explain the CS1 failure rate phenomenon, and no factor to date has been shown to influence programming performance across a range of different teaching contexts [27]. More recently, researchers have explored the relations between desirable aspects of programming behaviour and performance [23, 26]. But despite yielded promising results there is still no overall understanding as to why many students are able to program, whilst others endlessly struggle.

Many theories have been put forward as to why learning to program is a difficult task. Some are attributed to the nature of programming itself. Programming is not a single skill, but rather a complex cognitive activity, where a student must simultaneously build and apply several higher order cognitive skills to solve a particular problem [17]. Other reasons are attributed to aspects of the students. Students may lack motivation, they may be unable to create a mental model of how programs relate to the underlying system [2], or create a clear model of program flow [10].

Further reasons are associated with the teaching methodology used. Whilst many students fail programming courses, instructors face the additional challenge of adjusting their expectations to the students' level of ability [22]. These challenges have been acknowledged, and decades of research effort has been put into creating and applying teaching interventions that facilitate students' learning [17, 13]. These interventions can include moving from a traditional lecture and lab based approach to using pair programming, game based learning, or extreme apprenticeship.

However, to date, no study has attempted to quantitatively compare the impact that these approaches can have on improving the pass rates of failing programming courses. Without any quantitative evidence on the relative strengths of different approaches, the research community as a whole will continue to lack a clear consensus of precisely which methodologies provide the most effective means of teaching programming and saving failing programming students.

2. RESEARCH METHOD

The purpose of this study was to explore the degree to which various approaches of teaching programming could improve pass rates. In order to gather data for use in this study, a systematic review process was adopted, in an effort to identify as many articles that provided details of pre-intervention, and post-intervention pass rates as possible.

ICER '14 August 11 - 13 2014, Glasgow, United Kingdom

ACM 978-1-4503-2755-8/14/08 ...$15.00.
http://dx.doi.org/10.1145/2632320.2632349.

2.1 Research Questions

The questions answered in this study include:

1. How do teaching interventions reported in the literature increase students' success in CS1?
2. What practices do the successful teaching interventions comprise of?
3. Do so called best practices, or practices that are significantly better than others exist?

2.2 Identification of Relevant Literature

To identify approaches for teaching introductory programming and their influences on course success, an initial search of articles published between the years 1980 - January 2014 was carried out. Searches were made in the ACM and IEEE databases, after which further searches were made using Google Scholar in an attempt to identify both published and unpublished work which was not indexed by ACM and IEEE. A final search was conducted by manually screening the indexes of selected conference proceedings and journals for relevant studies, including: (1) Transactions on Computing Education, (2) SIGCSE, (3) ITiCSE, (4) ICER, (5) SIGITE, (6) ICALT.

Initial articles were selected based on keywords where boolean operators AND and OR were used to refine the searches. More specifically, the search criteria used was: (Improve OR Increase OR Decrease OR Lower OR Reduce) AND (Retention OR Attrition OR Pass OR Fail OR Success) AND (Programming OR Programming Course OR Introductory Programming OR CS1). Applying the criteria resulted in over 1000 somewhat relevant articles being identified, on which two researchers performed an initial inclusion screening. After applying an inclusion screening based on title and abstract, 226 articles remained. From these, further screening was made by including full text content, excluding articles that did not describe a replicable teaching intervention or an intervention that had overall been attempted at least twice, articles that did not discuss CS1 or introductory programming courses, articles that did not include pass-rates before or after the teaching intervention, and articles that did not include the amount of students before or after the teaching intervention (or data from which the numbers could be derived from). Finally, articles that described previously reported results from another perspective and articles from which all results were included in subsequent articles were excluded. The final number of articles used for this study was 32.

2.3 Study Coding

From each article, the following data was extracted:

1. Article details: publication year, name, author(s).
2. Course and institution details.
3. Teaching intervention: year, semester, used practices.
4. Totals and percentages before and after teaching intervention(s): n, pass, fail, withdraw, fail/withdraw.

All three authors performed extraction of details, at times extracting details from the same articles more than once. When conflicts occurred, the first author did an additional review.

From the 32 articles, 60 data entries with information on the amount of students as well as pass-rates before and after the intervention were extracted. When the intervention was done on a separate section during the same semester, the non-intervention and intervention results were paired. If there were multiple sections, an average combining all sections from the same semester was used to reduce possible instructor-related impact on the results. When results were described from subsequent semesters, the pairs were formed so that the pre-pair contained an average of all reported semesters before the intervention, and each post-entry contained details from one reported semester with the intervention. If an article described combined results from multiple institutions, it was included if the totals and percentages before and after the teaching intervention were included, and that the teaching intervention was described in detail.

To utilize a comparable measure of success, the reported totals and percentages in the articles were combined to describe a WDF-rate[1], i.e. the rate of students that did not withdraw, receive a D-grade[2] and did not fail the exam. The WDF-rate is a common measure used to describe course success, and it provides a more realistic view on success due to taking into account students that are not able to continue in their studies (D-grade in some institutions), students that fail in the exam, and students that withdraw from the course before the exam. When the WDF-rate was not readily available and the calculation WDF-grade was not possible, we chose the closest possible number assuming that the metric was the same in both pre- and post-intervention details. In this article, from now on we will use the term pass rate to describe WDF-rate or the closest number available.

We acknowledge that some institutions use a grace period during which students can drop out from the course without any sanctions. Unfortunately, very few articles did report such details, and thus it cannot be taken into account. Similarly, we acknowledge that the grading schema and learning objectives vary among different universities; unless otherwise noted in the articles, our assumption was that the learning objectives and grading schema remained similar between course instances, making the pre- and post-intervention details comparable. Additionally, we did not include demographic details or gender details into the study.

2.4 Classifying Teaching Interventions

Extracting teaching interventions was done in three phases. In the first phase the articles were coded based on the intervention types used. As an example, the article "Combining Cooperative Learning And Peer Instruction In Introductory Computer Science" [4] was coded with tags *cooperative learning*, *student group work*, *team teaching*, and *undergraduate teaching assistants*. The coding reflects the content; the article discusses collaboration and cooperative learning as the main activities, and the peer instruction discussed in the article describes the use of peer instructors, i.e. undergraduate teaching assistants, not to be confused with Peer Instruction by Eric Mazur. To give another example, the article "Experience Report: CS1 for Majors with Media Computation" [19] was coded with tags *media computation* and *peer instruction* as the article mentions that *"...one notable difference between the courses was that the media computation course*

[1]The acronym DFW was also visible in the literature; we utilize the acronym WDF.

[2]Typically 40-49% of the overall course score.

was taught using Peer Instruction in lectures, and the traditional course was not...". Although pair programming is also used, it is not coded as it is used in both non-intervention and intervention groups.

In the second phase, articles were supplemented with additional, descriptive tags. For example, for each article that was coded with the tag *media computation*, tags *content: media*, *contextualization*, *context: media*, were added to describe that the content (material) of the course was updated to contain media-type content, the course was contextualized so that the content had more meaning to the students that participated in the course, and finally, that the context revolved around media. Similarly, for each article that was coded with the tag *peer instruction*, tags *interactive classroom*, *student collaboration*, *reading before class*, *quizzes in class*, and *collaboration in class* were added. No limits were set on the amount of tags that could be used as long as the tags described the intervention properly. For example, an intervention where a game-themed final project was added to the course without additional modifications, the course was not contextualized, but the content was updated to include a game-theme component (tag *content: game-theme*).

Finally, in the third phase, equivalent or closely related tags were combined, and the changes were reflected to the article coding. At the end of the extraction phase, each article had the majority of original tags as well as a set of supplement tags that provided additional information on the described teaching activities. In total, 40 different tags remained after the combination phase, and each of the 60 data entries had on average 3.5 tags.

3. RESULTS

The results of the survey are analyzed from three different viewpoints. First, in Section 3.1 an overview to the results is provided, then in Section 3.2, the most common activities and their effect in the data are discussed, and finally in Section 3.3, the teaching approaches are analyzed based on a primary intervention type.

The results are considered in terms of *realized improvement*, i.e. the absolute improvement divided by the potential, by which the room for improvement that varies between different institutions is taken into account. For example, if a pre-intervention pass rate were 70% and post intervention pass rate were 85%, a potential change of 100-70 = 30% is available for the intervention. Of this, 15/30 or 50% was realized as the absolute improvement, or absolute percentage, was 15%.

3.1 Descriptive Statistics

Table 1 contains descriptive statistics of the data. On average, the pass rates before the intervention were 61.4%, and after intervention 74.4%. Much variance in both pre- and post-passrates exists; the smallest pass rate before intervention was 22.6% and 36% after intervention, while the largest pass rate was 94.2% before intervention, and 92.5% after intervention. The studied student populations varied also a lot. The smallest pre-intervention n was 15 students, which was from a targeted intervention to at-risk students, while the smallest post-intervention n was 9 students; the intervention stragegy in the study was applied to a small summer class. The largest number of students was 2298 before intervention, where the study reported data from past 16 iterations, and 1213 after intervention, which was from a study that reported aggregate results from multiple institutions.

descriptive	min	max	median	mean	sd (σ)
pass rate pre	22.6	94.2	63	61.4	15.5
pass rate post	36	92.5	74	74.4	11.7
students pre	15	2298	148	296.9	487.5
students post	9	1231	86	162.3	200.7

Table 1: Pass rates and study sizes before and after teaching intervention.

Five (8.3%) of the extracted data entries had a negative outcome (the pass rates decreased), while in 91.7% of the entries the intervention had at least a minor improvement on the overall results. On average, before the intervention, there was room for 38.4 percentage units of improvement, while after the intervention there was room for 25.6 percentage units. In other terms, the interventions improved the pass rates on average by 12.8 absolute percentage units, the realized improvement being 33.3% or nearly one third.

3.2 Overall Intervention Effect

Table 2 contains ten most frequent tags and the realized gains in the studies in which they appeared in. While the intervention types cannot be compared with each others due to overlapping, the table provides an overview of the realized improvements over different studies. The intervention tags encompass the following activities:

- *collaboration*: activities that encourage student collaboration either in classrooms or labs
- *content change*: at least parts of the teaching material was changed or updated
- *contextualization*: activities where course content and activities were aligned towards a specific context such as games or media
- *CS0*: the creation of a preliminary course that was to be taken before the introductory programming course; could be organized only for e.g. at-risk students
- *game-theme*: a game-themed component was introduced to the course, e.g. a game-themed project
- *grading schema*: a change in the grading schema; the most common change was to increase the amount of points rewarded from programming activities, while reducing the weight of the course exam
- *group work*: activities with increased group work commitment such as team-based learning and cooperative learning
- *media computation*: activities explicitly declaring the use of media computation (e.g. the book)
- *peer support*: support by peers in form of pairs, groups, hired peer mentors or tutors
- *support*: an umbrella term for all support activities, e.g. increased teacher hours, additional support channels etc.

When considering the median improvement, the studies that had media computation as one of the components were most successful, while studies with a game-theme were the least successful. Facilitating group work and collaboration,

intervention tag	n	min	max	median	avg	σ
collaboration	20	-1	59	39	34	17
content change	36	-17	69	34	34	17
contextualization	17	18	69	37	40	17
CS0	7	18	76	41	43	19
game-theme	9	-39	42	21	18	23
grading schema	11	3	42	30	29	12
group work	7	36	59	44	45	7
media computation	10	24	69	49	48	16
peer support	23	-1	59	36	34	16
support	9	-29	67	36	33	19

Table 2: Ten most common intervention tags and the overall intervention effects of the studies in which they appeared in. Number of studies including the intervention denoted as n, realized pass rates reported using minimum, maximum, median, average and standard deviation (σ) in percentages.

and creating a CS0 course were also among the high-performing activities. While the effect of an intervention activity depends naturally on other activities as well, a noticeable amount of variance was observed even within similar setups. The variance can be explained with the natural variance of student populations over different semesters, student intake, teacher effect, difference in grading criteria among different institutions, and the difference with student workloads among different institutions.

3.3 Primary Intervention Effect

Before comparing the impact of different interventions on programming pass rates, it was first important to determine whether there existed any significant differences in the pre-intervention pass rates of each intervention category, or whether the courses which were included in this study all had a comparable pre-intervention pass rate.

Grouping the 60 pre-intervention pass rates by the five primary intervention categories, a one-way ANOVA was performed. A Shapiro Wilk test confirmed the pass rates were normally distributed for all groups ($p > .05$), with the exceptions of relatable content and contextualization ($p = .01$), and hybrid approaches ($p = .01$). However as violations from normality do not substantially affect the type I error rate, and an ANOVA is considered relatively robust against this violation, we proceeded. Homogeneity of variances was confirmed by Levene's test ($p = .298$). A one-way ANOVA showed that there were no statistically significant differences in pre-intervention pass rates for the five primary intervention categories used in this study, $F(4,\ 55) = 2.17$, $p = .084$. To ensure that the violation of normality had not impacted the test result, we also performed a Kruskal-Wallis test, which confirmed that there were no statistically significant differences between the pre-intervention pass rates of each group, $\chi^2(4) = 9.13$, $p = .18$.

3.3.1 Collaboration and Peer Support

Approaches that include collaboration and peer support include peer-led team learning activities [9], pair programming activities [28] and cooperative and collaborative practices [4, 24]. Results are shown in Table 3. A total of 14 studies were classified as having applied an intervention, which primarily consisted of moving towards a collaborative, or peer support based approach. Three specific approaches were identified: cooperative learning (3 courses), team based learning (5 courses), and pair programming (6 courses). Out of all the interventions that were explored in this study, cooperative learning was found to yield the largest absolute improvement in CS1 pass rates (25.7% on average), and team based learning was found to yield the second largest absolute improvement (18.1% on average). Despite being frequently cited as an enabler for programming skills, the pair programming approach was only found to yield a absolute improvement of 9.6% on average, and ranked 11 out of the 13 interventions that were explored by this study. It was possible that courses to which this intervention was applied already had good pass rates, and therefore there was little scope for absolute improvement. When considering realized changes, we note that on average, pair programming yielded a realized increase of 27% in pass rates, but overall, this approach was still ranked 11th out of the 13 interventions which were explored by this study. Considering the results of all 14 courses combined, we found that instructors who applied a collaborative or peer support based intervention generally received the largest improvements in pass rates when compared to the other groups examined in this study (16.1% improvement, realized change 34.3%). A possible explanation is that the continuous feedback and cooperation from peers acts as an enabler for programming skills, supplementing feedback received from the compiler, which is not always at a sufficient level for inexperienced students to comprehend.

3.3.2 Bootstrapping

Bootstrapping practices either organized a course before the start of the introductory programming course [20, 7] or started the introductory programming segment using a visual programming environment such as Scratch and Alice [11]. Some of the activities were also targeted at at-risk students [16]. Results are shown in Table 4. A total of 9 studies were classified as having applied such an intervention. Two specific approaches were identified: using visual programming tools such as Scratch or Alice (5 courses), and introducing CS0 (4 courses). Out of all the interventions that were explored in this study, using visual programming tools were found to yield the fifth largest absolute improvement in pass rates (17.3% on average). A similar high ranking was found when considering realized improvement (fourth, 38.6%), which positioned using visual programming tools as the fourth overall best intervention. Whilst the absolute improvement for courses that introduced CS0 was much lower than visual programming (10.5% increase), the realized change that was yielded by this intervention was comparable (34.9% increase). Considering the results of all 9 courses combined, we found that instructors who applied a bootstrapping intervention generally received the second largest improvements in pass rates when compared to the other groups examined in this study (absolute change 14.3%, realized change 37.0%). It is possible that the initial simplification offered by these forms of intervention are able to assist students who might otherwise fail CS1, by suppressing the syntax barrier until they have gained sufficient knowledge of the underlying concepts. This also ties into research on threshold concepts, which suggested that reducing the level of complexity initially may be an effective way to assist students in overcoming thresholds.

3.3.3 Relatable Content and Contextualization

Approaches that introduced relateable content sought to make programming more understandable to students. These approaches include Media Computation [21], introducing real world projects [5] as well as courses that evolve around games [1]. Results are shown in Table 5. A total of 14 studies were classified as having applied an intervention, which primarily consisted of using relatable content and contextualization as a means to improve CS1 pass rates. Two specific approaches were identified: media computation (7 courses), and gamification (7 courses). Out of all the interventions that were explored in this study, using media computation was found to yield the seventh largest absolute improvement in pass rates (14.7% on average), and a comparable improvement was found for gamification (10.8% on average). However, when considering realized changes, media computation was found to yield the largest realized change across all interventions explored in this study (50.1% increase), whereas gamification was found to only yield the tenth largest (27.4% increase). Overall, and considering the results of all 14 courses combined, we found that instructors who applied a relateable content or contextualization intervention generally received the third largest improvements in pass rates when compared to the other groups examined in this study (absolute change 11.6%, realized change 38.7%). As media computation (overall rank 2) considerably outperformed gamification (overall rank 10), it could be the case that whilst games provide a useful tool to contextualize a learning task, there are still fundamental underlying programming concepts that can be better served by adopting a media computation approach.

3.3.4 Course Setup, Assessment, Resourcing

Approaches that modify course setup, assessment and resourcing included a broad range of practises starting from adjusting course content based on data from an assessment system [18], introducing new content, a programming tool that provides additional support and changing the grading schema assessment [15, 12]. Results are shown in Table 6. A total of 15 studies were classified as having applied an intervention which primarily consisted of changing aspects of the course setup, rather than changing elements of the teaching approach. Three specific approaches were identified: changing class size (4 courses), improving existing resources (2 courses), and changing assessment criteria (9 courses). Overall, the largest absolute improvements in pass rates were found by changing the class size (17.8% improvement) and improving existing resources (17.5%). However, when considering these improvements relatively, they were among the five worst interventions found by this study. Similarly, making changes to the assessment criteria applied in the course yielded on average an absolute improvement of 10.1% and realized improvement of 22.5%. But when considering these changes against the other 13 interventions explored by this study, changing assessment criteria ranked 12th. Considering the results from all 15 courses combined, we found that instructors who applied an intervention based on course setup generally yielded the fourth largest improvements in pass rates when compared to the other groups (absolute change 13.4%, realized change 26.8%). The findings on changing class size to improve pass rates are consistent with previous studies [3] that have suggested that smaller classes generally have lower failure rates than larger ones. However, overall, it is possible that this group of interventions were ranked as one of the lowest because making changes to the course setup, such as the assessment criteria, do nothing to adjust the likelihood of a student overcoming thresholds understanding programming concepts.

3.3.5 Hybrid Approaches

Hybrid approaches are approaches that upon discussion were not included in any of the primary categories. These include combinations of different practices [19, 14, 8]. Results are shown in Table 7. A total of 8 studies were classified as having applied an intervention, which primarily consisted of combining several different teaching interventions to yield a hybrid approach. Three combinations were identified: media computation with pair programming (2 studies), extreme apprenticeship (3 courses), and collaborative learning with relateable content (e.g. games) (3 courses). Overall, combining media computation with pair programming, or adopting an extreme apprenticeship approach were found to yield mid-range improvements in pass rates, ranging from 13.5-16.5% in absolute terms, or 36.9-49.3% in realized terms. These approaches were ranked fifth and seventh among the overall 13 interventions that were explored in this study. However, combining collaborative learning with content was found to be the worst overall intervention, actually yielding a decrease in pass rates of 9.7%, or 53.7% in realized terms. However, we note that some of the courses, which switched to this approach already had a very high pass rate ($> 90\%$), and therefore the scope for improvement was minimal.

3.3.6 Comparing Primary Interventions

The final question, which remained from this study, was to determine whether there were any significant differences in the post-pass rates of studies that applied different types of interventions. Grouping the 60 post-intervention pass rates by the five primary intervention categories, a one-way ANOVA was performed. A Shapiro Wilk test confirmed the pass rates were normally distributed for all intervention groups ($p > .05$) and homogeneity of variances was confirmed by Levene's test ($p = .487$). A one-way ANOVA showed no statistically significant differences in the post-intervention pass rates for the five primary intervention groups of this study, $F(4, 55) = 2.02$, $p = .105$. Similarly, a Tukey post-hoc analysis revealed no significant pairwise differences in post-intervention pass rates. This suggests that whilst substantial improvements in pass rates can be achieved by applying different interventions, the overall pass rates after applying different types of intervention are not substantially different.

4. DISCUSSION

The interventions reported in the literature increase introductory programming course pass rates by one third on average. A large part of the reported interventions increase student and teacher collaboration and update the teaching material and content in an attempt to make the content more relatable to the students. Support is facilitated in many ways; one approach is recruiting peer tutors that help students as they are working, while another approach is to build a CS0-course which acts as a bridge to the programming studies. Some interventions also changed the grading schema, which is known to affect students' behaviour. What may be missing however, are the reports on interventions

		Absolute Change			Realized Change			
Intervention	Courses	Mean	*SD*	Rank	Mean	*SD*	Rank	Overall
Cooperative	3	25.7	3.8	1 / 13	47.7	10.0	3 / 13	1 / 13
Team Based	5	18.1	11.6	2 / 13	35.0	12.3	6 / 13	3 / 13
Pair Programming	6	9.6	10.1	12 / 13	27.0	23.7	11 / 13	11 / 13
Overall Intervention	14	16.1	16.1	1 / 5	34.3	18.6	3 / 5	1 / 5

Table 3: Improvements in Pass Rates for Courses which applied Collaborative and Peer Support Interventions

		Absolute Change			Realized Change			
Intervention	Courses	Mean	*SD*	Rank	Mean	*SD*	Rank	Overall
Scratch and Alice	5	17.3	18.7	5 / 13	38.6	30.8	4 / 13	4 / 13
CS0	4	10.5	4.4	10 / 13	34.9	9.5	7 / 13	9 / 13
Overall Intervention	9	14.3	12.9	2 / 5	37.0	20.3	2 / 5	2 / 5

Table 4: Improvements in Pass Rates for Courses which applied Bootstrapping

		Absolute Change			Realized Change			
Intervention	Courses	Mean	*SD*	Rank	Mean	*SD*	Rank	Overall
Media Computation	7	14.7	5.4	7 / 13	50.1	18.9	1 / 13	2 / 13
Games	7	10.8	6.0	9 / 13	27.4	8.3	10 / 13	10 / 13
Overall Intervention	14	12.7	11.6	4 / 5	38.7	18.3	1 / 5	3 / 5

Table 5: Improvements in Pass Rates for Courses which applied Relatable Content and Contextualization

		Absolute Change			Realized Change			
Intervention	Courses	Mean	*SD*	Rank	Mean	*SD*	Rank	Overall
Class Size	4	17.8	16.6	3 / 13	34.0	43.2	8 / 13	6 / 13
Resource Improvement	2	17.5	2.8	4 / 13	32.1	5.2	9 / 13	8 / 13
Assessment	9	10.5	9.9	11 / 13	22.5	19.4	12 / 13	12 / 13
Overall Intervention	15	13.4	18.8	3 / 5	26.8	27.4	4 / 5	4 / 5

Table 6: Improvements in Pass Rates for Courses which applied Course Setup Interventions

		Absolute Change			Realized Change			
Intervention	Courses	Mean	*SD*	Rank	Mean	*SD*	Rank	Overall
Media Computation with Pair Programming	2	13.5	5.4	8 / 13	49.3	0.2	2 / 13	5 / 13
Extreme Apprenticeship	3	16.5	1.9	6 / 13	36.9	4.2	5 / 13	7 / 13
Collaboration with Games	3	-9.7	12.2	13 / 13	-53.7	67.9	13 / 13	13 / 13
Overall Intervention	8	6.0	13.2	5 / 5	6.0	61.6	5 / 5	5 / 5

Table 7: Improvements in Pass Rates for Courses which applied Hybrid Learning Approaches

that did not yield an improvement. Thus, educators that have tried an intervention but received poor results should also be encouraged and supported in reporting the results to create a more stable picture of the field.

Whilst no statistically significant differences between the effectiveness of the teaching interventions were found, marginal differences between approaches exist. The courses with relatable content (e.g. using *media computation*) with cooperative elements (e.g. pair programming) were among the top performers with CS0-courses, while courses with pair programming as the only intervention type and courses with game-theme performed more poorly when compared to others. However, these interventions were still able to improve pass rates by a minimum of 10%, suggesting that although they were not as strong as the other interventions in this study, they were still beneficial when compared to the traditional lecture and lab approach they replaced. Do note however, that many of the interventions did combine practices together, and e.g. the effect of a possible change in teaching material may be left unreported.

Nevertheless, the data confirms that educators and researchers are making a difference when trying out new teaching interventions and pedagogical approaches. One of the common denominator among all the interventions is change, while the other side of the coin – the "past situation" – may often be a state of complacency.

4.1 Related Work

While no such review exists from the field of introductory programming, many reviews on the influences of teaching approaches exist from other fields. The most notable one, a synthesis of 800 meta-analyses on teaching and achievement in schools by John Hattie [6], combines the results of multiple meta-analyses to form a picture of the efficiency of different teaching approaches. Although the results from Hattie's study are from schools, they provide an additional viewpoint to our findings. As an example, while collaborative and cooperative approaches were among the most effective approaches in this study, in schools the practices that require cooperation are above average in efficiency but increase in efficiency as students get older [6]. Similarly, peer tutoring in schools is not as efficient as in our study, which further provides support on the suggestion that the ability to support others and work in teams increases with age.

While many of the effects observed by Hattie were related to the teacher, such as the teacher clarity and instructional quality, little focus was on these aspects. The closest matches from our tagging are different feedback approaches and improving the course content; the first is also a part of the collaborative approaches.

4.2 Limitations

As the results presented in this article are derived from a synthesis of the results from other articles, a number of validity concerns, including the justification of the synthesis approach, can be raised which are discussed in this section.

Firstly, the teaching approaches that were used prior to the intervention were rarely explicitly stated in a very detailed fashion. For the most part, it was implied from the articles that the current approach was one based upon a traditional lecture and lab based approach. It is possible that in some cases important details were not reported in the articles, and thus were missed in the encoding process. Similarly, the learning objectives of the introductory programming courses were not considered for this study.

Secondly, the soundness of the teaching intervention design and experiments were not judged, and e.g. the quality of the used teaching material was not considered. It is likely that some interventions were implemented and executed better than others, which may lead to an imbalance in the results. Thus, one should not refrain from trying out the approaches that didn't seem to work.

Thirdly, the final number of selected articles, $n = 32$ is low, especially when considering that introductory programming courses have been studied for decades. A major concern in our case is the possibility of selective reporting. As only 8.3% of the studies contained negative results, it is possible that interventions with negative results have not often been reported. To counter this, we explored sources of *gray literature* via generalized searches, but our efforts were largely unsuccessful. The tendency to only encounter results for studies where an intervention has been successful however is a common limitation of systematic reviews.

Fourthly, whilst the results suggest that almost any planned intervention improves the existing state, the results have been gathered from studies that study university-level introductory programming courses. Further analysis should be performed when applying the results in other contexts to verify whether such approaches can yield similar improvements in pass rates at all levels of education.

Fifthly, the possible teacher effect and the effect of different student populations among different institutions is not taken into account. This is a deliberate choice due to the small amount of final articles.

Sixth, the choice for primary interventions and the tagging methodology has an inherent limitation as many of the studies had more than one intervention. Thus, the results shown here depend on the used classification, and different classification approaches may yield different results.

Finally, we note the unavoidable limitation that the assessment criteria of the individual courses were not the same over all data entries. Studies within the UK generally defined 'pass rate' as consisting of those students who had scored over 40% in the course. However, other studies defined 'pass rate' as consisting of those students who had scored at least a 'C', and others defined 'pass rate' as consisting of those students who had scored anything apart from an 'F'. Other studies did not supply details at all. Therefore this study unavoidably has to assume that a consistent notion of 'pass rate' exists and holds valid across the different teaching contexts. However, we note that this is a common limitation of other studies of this nature (e.g. [3, 25]).

5. CONCLUSION

In this article, we performed a quantitative systematic review on articles describing introductory programming teaching approaches, and provided an analysis of the effect that various interventions can have on the pass rates of introductory programming courses. While the total amount of articles was relatively low, a total of 60 pre-intervention and post-intervention pass rates, describing thirteen different teaching approaches, were extracted and analyzed.

The results showed that on average, teaching interventions can improve programming pass rates by nearly one third when compared to a traditional lecture and lab based approach. While no statistically significant differences be-

tween the effectiveness of teaching interventions were observed, marginal differences do exist. The courses with relatable content (e.g. using *media computation*) with cooperative elements (e.g. pair programming) were among the top performers with CS0-courses, while courses with pair programming as the only intervention type and courses with game-theme performed more poorly when compared to others.

What the results of this analysis mean in practice, is that educators and researchers that are applying teaching interventions are making a difference. Whilst there is no silver bullet, no teaching approach works significantly better than others, a conscious change almost always results in an improvement in pass rates over the existing situation.

6. REFERENCES

[1] J. D. Bayliss. The effects of games in CS1-3. In *Microsoft Academic Days Conference on Game Development in Computer Science Education*, pages 59–63. Citeseer, 2007.

[2] M. Ben-Ari. Constructivism in computer science education. In *SIGCSE bulletin*, volume 30, pages 257–261. ACM, 1998.

[3] J. Bennedsen and M. E. Caspersen. Failure rates in introductory programming. *SIGCSE Bulletin*, 39(2):32–36, 2007.

[4] J. D. Chase and E. G. Okie. Combining cooperative learning and peer instruction in introductory computer science. *SIGCSE Bulletin*, 32(1):372–376, Mar. 2000.

[5] N. De La Mora and C. F. Reilly. The impact of real-world topic labs on student performance in CS1. In *Proc. Frontiers in Education*, pages 1–6. IEEE, 2012.

[6] J. Hattie. *Visible learning: A synthesis of over 800 meta-analyses relating to achievement*. Routledge, 2013.

[7] M. Haungs, C. Clark, J. Clements, and D. Janzen. Improving first-year success and retention through interest-based CS0 courses. In *Proc. SIGCSE*, pages 589–594. ACM, 2012.

[8] J. Kurhila and A. Vihavainen. Management, structures and tools to scale up personal advising in large programming courses. In *Proc. SIGITE*, pages 3–8. ACM, 2011.

[9] P. Lasserre and C. Szostak. Effects of team-based learning on a cs1 course. In *Proc. ITiCSE*, pages 133–137. ACM, 2011.

[10] I. Milne and G. Rowe. Difficulties in learning and teaching programming - views of students and tutors. *Educ. and Information Technologies*, 7(1):55–66, 2002.

[11] P. Mullins, D. Whitfield, and M. Conlon. Using alice 2.0 as a first language. *Journal of Computing Sciences in Colleges*, 24(3):136–143, 2009.

[12] U. Nikula, O. Gotel, and J. Kasurinen. A motivation guided holistic rehabilitation of the first programming course. *Trans. Comput. Educ.*, 11(4):24:1–24:38, Nov. 2011.

[13] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson. A survey of literature on the teaching of introductory programming. In *SIGCSE Bulletin*, volume 39, pages 204–223. ACM, 2007.

[14] L. Porter and B. Simon. Retaining nearly one-third more majors with a trio of instructional best practices in CS1. In *Proc. SIGCSE*, pages 165–170. ACM, 2013.

[15] D. Radošević, T. Orehovački, and A. Lovrenčić. New approaches and tools in teaching programming. In *Proc. of Central European Conference on Information and Intelligent Systems*, pages 49–57, 2009.

[16] M. Rizvi and T. Humphries. A scratch-based cs0 course for at-risk computer science majors. In *Proc. Frontiers in Education*, pages 1–5. IEEE, 2012.

[17] A. Robins, J. Rountree, and N. Rountree. Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172, 2003.

[18] S. C. Shaffer and M. B. Rosson. Increasing student success by modifying course delivery based on student submission data. *ACM Inroads*, 4(4):81–86, Dec. 2013.

[19] B. Simon, P. Kinnunen, L. Porter, and D. Zazkis. Experience report: CS1 for majors with media computation. In *Proc. ITiCSE*, pages 214–218. ACM, 2010.

[20] R. H. Sloan and P. Troy. CS 0.5: A better approach to introductory computer science for majors. *SIGCSE Bulletin*, 40(1):271–275, Mar. 2008.

[21] A. E. Tew, C. Fowler, and M. Guzdial. Tracking an innovation in introductory CS education from a research university to a two-year college. In *Proc. SIGCSE*, pages 416–420. ACM, 2005.

[22] I. Utting, A. E. Tew, M. McCracken, L. Thomas, D. Bouvier, R. Frye, J. Paterson, M. Caspersen, Y. B.-D. Kolikant, J. Sorva, and T. Wilusz. A fresh look at novice programmers' performance and their teachers' expectations. In *Proc. ITiCSE Working Group Reports*, pages 15–32. ACM, 2013.

[23] A. Vihavainen. Predicting students' performance in an introductory programming course using data from students' own programming process. In *Proc. ICALT*, pages 498–499. IEEE, 2013.

[24] H. M. Walker. Collaborative learning: a case study for CS1 at grinnell college and austin. In *SIGCSE Bulletin*, volume 29, pages 209–213. ACM, 1997.

[25] C. Watson and F. W. Li. Failure rates in introductory programming revisited. In *To appear in Proc. Innovation and Technology in Computer Science Education (ITiCSE)*. ACM, 2014.

[26] C. Watson, F. W. Li, and J. L. Godwin. Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *Proc. ICALT*, pages 319–323. IEEE, 2013.

[27] C. Watson, F. W. Li, and J. L. Godwin. No tests required: comparing traditional and dynamic predictors of programming success. In *Proc. SIGCSE*, pages 469–474. ACM, 2014.

[28] L. Williams, C. McDowell, N. Nagappan, J. Fernald, and L. Werner. Building pair programming knowledge through a family of experiments. In *Proc. Empirical Software Engineering*, pages 143–152. IEEE.

APPENDIX

Due to space limitation and to serve as a starting point for future researchers, a list of 32 references is provided at http://bit.ly/1qkb8GI.