Relationship between perceived problem-solving skills and academic performance of novice learners in introductory programming courses

Ashok Kumar Veerasamy[1], Daryl D'Souza[2], Rolf Lindén[1] and Mikko-Jussi Laakso[1]

[1] Department of Future Technologies, University of Turku, Turku, Finland

[2] School of Computer Science and Information Technology, RMIT University, Melbourne, Australia

Corresponding author: Ashok Kumar Veerasamy, Department of Future Technologies, University of Turku, Turku, Finland, e-mail: ashok.veerasamy@utu.fi.

Abstract

Past research has shown that student problem-solving skills may be used to determine student final exam performance. This study reports on the relationship between student problem-solving skills[1] and academic performance in introductory programming, in formative and summative programming assessment tasks. We found that the more effective problem-solvers achieved better final exam scores. There was no significant difference in formative assessment performances between effective and poor problem-solvers. It is also possible to categorize students based on problem-solving skills, in order to exploit opportunities to improve learning around constructivist learning theory. Finally, our study identified transferability skills and the study may be extended to identify the impact of problem solving transfer skills on student problem solving for programming.

Keywords: Problem-solving in programming; assessment tasks; learning transfer; problem-based learning constructive alignment

1. INTRODUCTION

Problem solving is a valuable and desirable skill if one is to be successful in learning and in the workplace. (Voskoglou & Buckley, 2012; Argaw et al., 2017). Confidence in one's problem-solving

[1] Henceforth, for brevity, we drop the word "perceived" in "student perceived problem-solving skills" and use either "student perceived problem-solving" or simply "student PSS"

ability might influence student performance in assessment tasks (Bandura, 1977; Lishinski, Yadav, Enbody, & Good, 2016). Research in computer science education has highlighted that many novice students lack problem-solving and computational thinking skills and have difficulties in utilizing key programming concepts to express their solutions in code (Lister, et al., 2004; Koulouri, Lauria, & Macredie, 2015; Uysal, 2014). Hence, it is important to examine the relationship between student problem-solving skills (PSS) and their performance in introductory programming; early awareness of students' problem-solving abilities allows for strategically assisting students to further develop such skills and their programming skills. As an aside, research in computer science education has examined the prerequisite factors important in predicting student performance (Longi, 2016; Veerasamy, Daryl D'Souza, & Laakso, 2018). However, in spite of research on factors that contribute to success in programming, a key question that is often asked is: *Why is learning to program is easier for some than the others?* The research reported here aims to determine whether student PSS is relevant to student performance in learning programming. Our interest was motivated by the increasingly prevailing presence of students entering our first-year programming course with varied PSS or experience, and the need to develop inclusive teaching strategies to engage students. Towards this objective, we pose and address the following research questions:

a) Is perceived problem-solving skills related to student performance in ongoing assessment tasks?

b) Is perceived problem-solving skills related to student performance in the final programming exam?

c) Is it possible to propose the student problem-solving skills as a predictor to predict student performance in final programming exam?

The paper is organized as follows. Section 2 presents a literature review of studies conducted around PSS, its impact on assessment tasks, and its significance in relation to learning programming and student final exam scores. Section 3 describes our research methodology. Section 4 presents the findings of the study, which we discuss in depth in Section 5. Finally, Sections 6 and 7 present our

conclusions, limitations, future work and pedagogic implications; also, we identify some related future work directions, to develop a more enhanced and innovative approach to teaching introductory programming courses. The scope of the present study is limited in the investigation of relationship and effects of cognitive factor PSS, on student academic performance in selected assessment tasks. The realtionship and effects of other causal factors on student academic performance is beyond the scope of the present study and will be dealt with in the future study.

2. RELATED WORK

Problem solving is a metacognitive skill, which reveals the way a person learns and experiences different aspects of the problem-solving process; it is a learned life skill and every individual has their own problem solving abilities, learned at different paces through various situations in daily life (Dostál, 2015; Ozus et al., 2015). However, an individual's problem-solving skill is related to their problem awareness ability, perceptibility of the problem, willingness to solve the problem, competency to solve the problem, and cognitive self-evaluation (Dostál, 2015). Self-efficacy influences how well students approach problems (Askar, 2009), implying that self-efficacy in learning is one of the motivational components of problem-solving (Eskin, 2013). Several studies have emphasised the importance of PSS (Md.Yunus et al., 2006; Voskoglou & Buckley, 2012; Argaw et al., 2017). Moreover, PSS is listed by employers as a desirable, generic soft skill (Md.Yunus et al., 2006; White et al., 2013; Kappelman et al., 2016). In addition, problem-solving is listed as one of the key skills to study of computer engineering and information technology (Sabin et al., 2017), and as crucial, valuable skill for both novice and qualified IT professionals (Kappelman et al., 2016).

Programming is a complex activity, especially for novices; it requires certain cognitive skills as a prerequisite (Pea & Kurland, 1984). Learning to program requires the learner to think, understand the general concepts of the program, and general problem-solving abilities to analyse, organize, implement

and evaluate the code outcomes (Pea & Kurland, 1984; Falloon, 2016). Several studies have examined the importance of PSS in learning programming, and explored the integrated techniques, and factors that promote PSS in introductory programming courses (O'Grady & J, 2012; Uysal, 2014; Koulouri, Lauria, & Macredie, 2015; Chao, 2016). For example, while problem-based learning fosters students' critical thinking, the presence of problem-based learning in computing curricula is not deep (O'Grady & J, 2012). Despite the mixed results studies suggest that students who have problem-solving competencies learn and perform better in programming and that learning programming improves student higher-order thinking, self-efficacy and PSS (Tu & Johnson, 1990; Psycharis & Kallia, 2017; Yukselturk & Altiok, 2017). In addition, teaching problem solving before programming improves programming performance (Koulouri, Lauria, & Macredie, 2015). However, most novice programming students have difficulties in formulating a problem and expressing its solution in code. Furthermore, students who lack PSS have difficulty in utilizing key concepts, such as loops and conditionals in programming (Koulouri, Lauria, & Macredie, 2015). These studies suggest students' PSS may influence their learning in programming courses and that there is a link between student PSS and learning programming.

Educators use formative assessments to measure student learning progress in order to ascertain learning difficulties and recommend remedial methods to improve student learning (Trumbull & Lash, 2013). However, students choose assessment tasks that they believe they are capable of completing otherwise they tend to avoid those tasks. Moreover, confidence in one's problem-solving ability influences how long they will persist in difficult tasks and there is a correlation between student problem-solving ability and performance in programming assignments (Bandura, 1977; Lishinski et al., 2016) suggesting that problem-solving ability may influence student performance in assessments.

Prerequisite factors are important in predicting student performance (Longi, 2016; Veerasamy, Daryl D'Souza, & Laakso, 2018). Consequently, the search continues for valuable predictor(s) of

student performance. However, to our knowledge no previous study has used a concrete inventory for use as a possible predictor and predictor variables used in the studies has varied from one context to another in several ways, including student cohort, cultural setting, etc. (Erdogdu & Erdogdu, 2015; Sharma & Shen, 2018) Several studies have been conducted on the impact of PSS on student performance in various courses including programming (Shrout & Witty, 1990; Adachi & Willoughby, 2013; Bester, 2014; Lishinski et al., 2018). Heppner et al. reported appraising student PSS may help instructors to identify student study habits and attitudes that are important to academic performance, though it is theoretically unrelated to academic achievement (Heppner & Petersen, 1982). Omiwale conducted a study to identify the relationship between problem-solving ability and achievement in physics among senior secondary school students, and concluded that students with better PSS get higher grades in physics (Omiwale, 2011). Shrout et al. study reported that student problem-solving appraisal and academic achievement can be a significant predictor for course grade (Shrout & Witty, 1990). Similarly, Bester examined the relationship between problem-solving proficiency of sophomore mathematics students and a quantitative techniques course and reported that there is a strong relationship between students' problem-solving proficiency and their achievement in quantitative techniques course (Bester, 2014). Lee et al. study has revealed that lack of cognitive strategies in problem-solving impacts student performance in programming (Lee & Thompson, 1997). In addition, Nowaczyk et al. concluded that testing a student's prior PSS in the beginning of the programming course would help to predict student programming performance (H.Nowaczyk, 1984).

In addition, several studies reported that gender and initial self-efficacy differences did not impact novice programming learning outcomes (Bubica & Boljat, 2014; Akar & Altun, 2017; Lishinski et al., 2018). Specifically, our Independent Samples *t*-Test results confirmed that the PSS and final programming exam mean scores between male and female students is not significantly different. These

aforementioned studies and preliminary statistical results suggest that gender differences do not have significant effect in programming learning.

Overall, our contribution is novel in that it focuses on finding the relationships between PSS and assessments and to determine if PSS could be a useful predictor of performance.

3. RESEARCH METHODLOGY

The aim of this study was to examine the relationship between PSS and the results of formative and summative assessment in programming. We measured the correlation between PSS and student performance in an introductory programming course (see below). Data was collected in one semester (2016), from nearly 200 enrolled students. Of these 166 students completed the problem-solving inventory (PSI), used to ascertain students' PSS. They also completed homework and demo exercises, as well as an electronic final examination.

We used Spearman's Rank correlation coefficient technique to describe the relationship between non independent variables such as PSS scores and selected assessment tasks, including the final programming exam. Being mindful that correlation does not imply causality we nevertheless proceeded with causation experiments for a better understanding of relationships between the variables. We used the Kruskal-Wallis and the Bonferroni ad hoc tests to test for if any statistical significant differences in academic performance between students with high and low PSS scores. There were other formative assessment components such as project work, which was included to calculate final scores for this course. However, project work was done in groups of two or three students, so project work was not considered in our analysis.

3.1 Description of the course and data collection

Algorithms and Programming has approximately 150-200 students enrolled in the autumn semester of each year. The course comprises contact hours (28 hours of lectures, 8 hours of tutorial sessions) and

non-contact hours, approximately 100 hours for independent work, including demo, homework, practice exam and discussion of project or assignment work, over the 8 weeks of semester. The following subsections present assessment details, which include homework exercises, demo exercises, and the final exam.

3.1.1 ViLLE: e-learning tool

Part of the course assessment used the ViLLE software tool (Rajala & Erkki Kaila). ViLLE was used by students for homework and class demonstration sessions, and the final exam. ViLLE is a custom learning environment that facilitates development of share learning materials and assessment tasks. It supports automatic grading, file submission and allows instructors to manually assess submitted work. ViLLE has been used as a collaborative education tool and shown to be effective to improve student motivation and performance (Laakso, Kaila, & Rajala, 2018). In addition, online exams and surveys may be presented via ViLLE, to measure student performance and skills. We used ViLLE to survey students to determine prior programming knowledge in an earlier study (Veerasamy, Daryl D'Souza, & Laakso, 2018). At the time of writing, ViLLE has been used by 6735 instructors of various courses including mathematics (8189), Finnish language (3236) and programming (920), at school and university levels. Over 132738 students have registered with ViLLE to date, for online learning support.

3.1.2 Formative assessment: ViLLE Homework (HE) and Demo exercise (DE)

Homework for *Algorithms and Programming* is set weekly for 8 weeks. Each set of exercises has 5-10 questions, comprising exercise types: objective, code tracing, visualization, filling missing parts of code. A demo exercises are set bi-weekly, after the first three weeks of the semester. Each set has 4-7 coding questions. The maximum possible total score for HEs is 217 and for DEs, 400. The due date for homework is usually one week after the HE notification date. Students are allowed to submit their

answers as many times as they like (via ViLLE), each new submission replacing previous submissions. Submitted answers for HE are automatically graded. Students receive instant feedback/scores for every submission. While instructors may view submission history, students can no longer see previous submissions. The purpose of "multiple submission attempts" and feedback is to support student learning and their study behaviours, and to improve their scores. A few code completion exercises are manually graded by instructors and marks with feedback entered via ViLLE. Figure 1 presents a screenshot illustrating student interaction with ViLLE.

----------Figure 1 here----------

DE exercises are also delivered to students via ViLLE a week before the day when DE sessions are to be conducted. Figure 2 shows a sample screenshot of coding exercises for a DE session.

----------Figure 2 here----------

Students are expected to prepare DE solutions at home and present their solutions to designated DE sessions. In a DE session, all students' solutions are discussed, and a few students are selected randomly via ViLLE, to demonstrate their answers to the entire class. No marks are awarded for class demonstrations (DEs). However, students who complete the DE exercises are instructed to enter their solutions into ViLLE, via the instructor's computer for subsequent marking. The marks for DEs are calculated by ViLLE based on their registered responses in the lecturer's computer (Veerasamy, et al., 2016). Both HE and DE are hurdles and students should secure at least 50% in each category to be eligible to sit final exam.

3.1.3 Summative assessment: Final Exam (FE)

There is one summative assessment task, the final examination (FE), which is conducted at the end of the course of study, electronically via ViLLE. The exam duration is 180 minutes. The FE is a hurdle and students must secure at least 50% to pass the course. The exam is divided into three sections:

multiple choice questions, short answer questions, and coding questions. The maximum possible score for the FE is 90.

3.1.4 Problem-solving Inventory (PSI)

The PSI is a questionnaire to measure an individual's self-appraisal in problem-solving skills (Heppner & Petersen, 1982). The PSI questionnaire contains 35 closed-format Likert type questions including three filler questions with 6-point in the Likert scale. The PSI consists of three subscales denoted: PSC (11 items), AAS (16 items) and PC (5 items) (Table 1).

----------Table 1 here----------

The PSI questions (in English), were translated into Finnish for students whose native language is Finnish and validated by colleagues (fluent in English and Finnish) for linguistic quality checking and equivalence. The PSI questions were presented via ViLLE at the beginning of the semester, to be completed optionally. The reliability and validity of PSC is .85; AAS is .84, PC is.72 and the total inventory (N =150) is .90, which suggests that the constructs were internally consistent. Similarly, the estimates of validity for PSI suggest that the scores of three factors (PSC, AAS and PC) are significantly correlated (ps < .0001) with students' ratings of their levels of PSI, and satisfaction/dissatisfaction with their PSI (Heppner & Petersen, 1982). In addition, the Cronbach Alfa internal consistency reliability coefficient of the PSI scale is .88 and the credibility coefficient obtained with dividing in half was found as r = .85 (Akben, 2018). It shows that PSI has very good internal consistency and stability in predicting an individual's self-perception of PSS. This PSI has also been used to measure student PSS in programming courses (Yurdugül & Aşkar, 2013; Uysal, 2014; Özen, 2016). We ran the Cronbach alfa, a psychometric test to measure PSI reliability yielding 0.855, which indicates a high level of internal consistency for our scale, with the collected data. The validity of the survey responses based on student honesty; students tend to respond selectively to overweight their

own capabilities in order to present a good impression (Rosenfeld, Booth-Kewley, & Edwards, 1996). However, studies have vouched for PSI's significant internal consistency in predicting student PSS.

4. DATA ANALYSIS AND RESULTS

Table 2 provides the PSS score bands and the number of students by PSS levels calculated via the PSI survey, and as defined by Heppner et al. for further analysis (Heppner & Petersen, 1982).

----------Table 2 here----------

A Shapiro-Wilk test was conducted to check if data were normally distributed (Ghasemi & Zahediasl, 2012). The p-value was smaller than the alpha level. That is, the mean scores for PSS, HE, DE and FE were not normally distributed. Accordingly, we used Spearman's Rank correlation coefficient (SpR) to measure the statistical dependence between the selected variables PSI and HE, DE, and FE (Jauke & Kossowski, 2011). Table 3 presents the summary of SpR results between the assessment task variables, for the year 2016.

----------Table 3 here----------

There is a negative linear relationship between PSS and HE, DE and FE. The Sig values (2-tailed) of HE (0.034), DE (0.016), and FE (0.001) point to a correlation between the selected variables and hence evidence to reject the null hypothesis of no relationship between the two variables. In addition, the negative correlation values indicate that students who have low PSS may struggle to perform better in FE. However, the SpR correlation values of PSS and HE (-0.165) and PSS and DE (-0.199) are weak although the SpR correlation value of PSS and FE (-0.254) is nearly moderate. As mentioned, students who have poor PSS will also perform poorly in formative assessment tasks. We verified this via the average HE and DE scores for each PSS level to ascertain if there was a significant difference in assessment tasks performance between PSS levels. Table 4 presents the results.

----------Table 4 here----------

On average, students with Level1 PSS performed better in assessment tasks than students with lower PSS (Levels 2 and 3). The average formative assessment (HE and DE together) score for PSS Levels 1, 2 and 3 is, respectively, 82%, 79% and 75%.

----------Figure 3 here----------

We used the Kruskal-Wallis test to show that there was no statistically significant difference in HE scores between the different PSS groups ($\chi^2(2) = 1.350$, $p = 0.509>0.05$), with a mean rank HE score of 92.42 for PSS Level1, 83.26 for PSS Level2 and 79.04 for PSS Level3. Similarly, a corresponding Kruskal-Wallis test for DE yielded a p-value $0.173 > 0.05$. Hence, there was no statistically significant difference in DE scores between the different PSS levels ($\chi^2(2) = 3.507$), with a mean rank DE scores of 94.44, 85.38 and 73.73, respectively for PSS Level1, Level2 and Level3. As the test results were insignificant, we did not do a Bonferroni's post-hoc correction on HE and DE for students with different PSS.

Similarly, students who are good problem solvers performed significantly better in the final examination. To answer the second research question (b), we calculated the average FE scores for each PSS level, to ascertain if there was any significant difference between the three PSS groups, based on their overall mean FE scores. Table 5 and Figure 4 reveal a significant difference between PSS levels for average FE scores (83%, 79% and 69%). Overall, students with effective and moderate PSS (Levels 1 and 2) secured higher scores than students with poor PSS (Level3) in the final programming examination.

----------Table 5 here----------

----------Figure 4 here----------

We also identified the impact of PSS on FE scores (out of 90 marks) to answer the research question (b). Figure 5 presents the number (%) of students versus score range for various PSS levels.

----------Figure 5 here----------

23% of students at PSS Level3 attained low scores (<45); however, only 11% of students at PSS Level1 and 12% of students at PSS Level2 got low scores in the FE. Nearly 11% at Level3, 20% at Level2, and 8% at Level1, got 67-77 marks in the FE. Also, 50% Level1 got above 88-90 marks in the FE. However, only 8% at Level3 got above 88-90 marks in the FE. These differences answer research question (b) (that student PSS may influence student final programming exam performance). Furthermore this was confirmed by a Kruskal-Wallis test followed by Bonferroni post-hoc correction. There was a statistically significant difference in FE scores between the different PSS groups ($\chi^2(2) = 11.700$, $p = 0.003 < 0.05$), with a mean rank FE scores of 101.46 (Level1), 87.98 (Level2) and 64.69 (PSS Level3). In addition, Figure 6 presents the overall grades obtained by all students despite their PSS levels.

----------Figure 6 here----------

In addition, the post hoc test using the Bonferroni correction also revealed that the distribution of FE scores for PSS Level3 was statistically significantly different to the FE scores for PSS Levels 1 ($p = 0.026$) and 2 ($p = 0.029$). However, the distribution of FE scores for PSS Level2 is not statistically significantly different to FE scores for PSS Level1 ($p = 1.000$). Therefore, we conclude that there is a statistically significant difference in FE scores between the students with high-level PSSs compared with those with low-level PSS. Our Bonferroni multiple comparison post-hoc correction results prompted us to ascertain whether PSS can serve as a significant predictor for student achievement in final exams for programming (research question (c)). We used multiple linear regression analysis to identify whether or not FE would be impacted by PSS.

----------Table 6 here----------

The multiple regression results (Table 6) clearly demonstrate that PSS may be used as a good regression model variable to predict FE (p-value ($0.004 < 0.050$, coefficient 0.222) is significantly

different from 0. Hence, student PSS scores can be included as predictor variables for the purposes of developing a predictive model to predict students' final exam performances in introductory programming courses.

5. DISCUSSION

This study investigated the relationship between student problem-solving abilities and their performance in formative and summative assessment tasks in an introductory programming course. The foregoing analysis revealed a monotonic relationship between student PSS scores and formative and summative performances. However, the strength of the relationship between these variables (HE, DE, and FE) on PSS is weak (Table 3), implying that PSS may impact academic performance in programming courses. Furthermore, there is no significant difference in performance in formative assessments between students with effective PSS and students with poor PSS (Figure 3). Hence, these results do not answer our research question (a) although there is a small mean (HE and DE mean scores) difference in formative task scores among the various PSS groups (Table 4).

The SpR results for PSS and FE (Spr = -254) and mean FE results suggest that effective problem solvers may perform better in the FE than poor problem solvers (Table 5). Specifically, the results of multiple comparison tests (Kruskal-Wallis and Bonferroni correction) answer our research question (b), revealing that there is a statistically significant difference in final exam scores between the students with effective PSS and those with poor PSS, although both groups of students performed similarly on formative assessment tasks. In addition, Figure 5 results suggest that students with poor PSS may struggle to achieve high scores in FE compared to students with effective and moderate PSS. Furthermore, we found no significant differences between effective problem solvers and moderate problem solvers in their FE scores. However, Table 6 results answer our research question (c) and suggest that student PSS could be a possible predictor for student achievement in final exams of

programming. In addition, the findings on relationship between student PSS and FE performance is consistent with literature review of other studies (Omiwale, 2011; Bester, 2014). However, our study results did not congruence with research studies examined the relationship between student prior PSS and final grades (Özyurt, 2015; Lishinski et al., 2018). Therefore, we conclude that the students with poor PSS scored on average lower than those with good and effective PSS.

The multiple linear regression results for PSS on FE scores, and the differences between mean FE scores within each PSS group answered research question (c), and suggested that student PSS may impact FE performance. Therefore, we conclude that student PSS can be considered as a significant factor to predict student final programming exam performance. However, our research findings on relationship between PSS and formative assessment tasks led us to surmise that in general PSS may not be considered as a significant factor to predict student performance in formative assessment tasks, although this should be analysed further to support our conclusion. In addition, the results on connection of PSS on formative assessment tasks raised a few other controversial points. First, increasing use of the Internet and other support systems as resources for solution ideas which, in turn, may have caused a diminution of originality in analysing problems and in problem solving; students often use ICT applications to obtain results for given assessment tasks (Veerasamy & Souza-Daw, 2012). Second, formative assessments are designed to assess students for learning, and to improve learning outcomes and PSS. However, if the student is not genuinely involved or does not invest the required amount of effort in completing and submitting formative assessment tasks, then it is difficult for those students to perform better in the FE. This is because, typically, final exams assess overall student knowledge of the subject, requiring students to think critically, to find solutions. For example, students are often are presented problems in final exams, previously not encountered. However, students are expected to solve these new problems through problem-solving techniques that they learnt via formative assessments and other learning sources (Martin, 1971). This implies that if the students

have not developed a deep learning, via their formative assessments, to prepare solutions, they may struggle to perform well in summative tasks. Moreover, if the instructional methodologies and formative assessments are not aligned with summative assessments, then students may struggle to cope with final exam stress or may be unable to transfer their problem-solving skills (Mayer, 1998). That is, students' lack of "learning transferability" requires application of what is previously learned to intended assessments. For example, if students are presented with problems in summative assessment tasks that are not previously encountered in formative assessment tasks, during or prior to their study, they are likely perform poorly or even fail. In addition, if the students fail to solve formative assessment problems, they too may have issues related to PSS transferability, when attempting to solve non-routine problems. As noted, solving a programming problem requires computational thinking, which is a subset of PSS, and is considered to involve abstraction, automation, solution execution and evaluation, including the act of transferring existing knowledge for new situations (Voskoglou & Buckley, 2012). It is possible that formative assessment tasks used in this course were not as well aligned, if at all, with the FE questions or students were not introduced to techniques to solve non-routine problems. Furthermore, we also infer from our Kruskal-Wallis test results on PSS and formative assessment tasks that it is possible that students might have correctly completed the given formative assessment, without understanding the objective of the formative assessment question (Jolliffe, 1990). However, for formative exercises not discussed in the class, students may well have felt justified in using other resources to complete homework. In addition, it is also possible that students might have not done formative assessments on their own, even though they were encouraged to use multiple submission attempts and, in turn, to develop their PSS. This needs further analysis. Despite these results, it raises the question: *What can be done to help students who do not have problem solving transferability skills despite understanding how to solve routine problems presented in assessment tasks?* Our results (Table 4 and Figure 3) revealed that poor problem solvers performed

similarly to moderate and effective problem solvers in formative assessment tasks. However, they failed to achieve high scores in the final exam, due to lack of problem-solving transferability skills or lack of familiarity with summative assessment tasks (Table 5). Therefore, further analysis is required to identify the similarities and differences between students with different PSS levels based on problems presented in formative assessment tasks and summative assessment tasks. However, in keeping with several previous studies, our results (Table 6) revealed that PSS has a marked effect on student learning outcomes, and is one of the strongest determinants of summative performance indicators for programming courses. Hence, PSS may be used to determine student learning and performance.

6. CONCLUSION, LIMITATIONS AND FUTURE WORK

We have identified that performance in formative assessment and PSS are weakly correlated. However, student PSS and FE scores have a moderately negative correlation. Specifically, there is a difference in FE scores between students with good versus those with poor PSS. Additionally, students with poor PSS may have issues with problem-solving transferability skills, which need further study. Therefore, our results provide evidence that existing assessment tasks in introductory programming may need changes in order to bridge the gap between PSS and performance in assessments. It may be concluded that measuring student PSS in the beginning of novice programming course can be useful in predicting the student final programming exam performance in the course. In addition, our results represent a motivation to ascertain factors that prevent students with poor PSS from securing high scores in FE.

Our study has several limitations. First, the sample size was not sufficiently large and the data were obtained from one course within one university. Second, we used self-reported survey data to examine student PSS levels, which may contain potential sources of bias; it is unknown whether or not students responded to the questionnaires, honestly. Despite these limitations, our findings provide some further ideas for both teaching practice and future research. Both PSS and assessment tasks are important

variables with PSS being strongly connected with student final exam performance. Hence, establishing student PSS at the outset may aid instructors to obtain pedagogically meaningful information to allow for strategies to alleviate problem-solving.

Our study may be extended to identify the impact of problem-solving transferability skills on student programming problem-solving, by examining the following questions: *How to improve student programming problem-solving skills via assessment tasks? What may be done to help students to be creative when they are faced with non-routine programming problems? How student non-routine problem-solving skills may be promoted by providing programming assessment tasks? What is the impact of student problem-solving transferability skills on student performance in programming courses? How do student general problem-solving skills differ from specific problem-solving skills for learning programming?*

7. EDUCATIONAL IMPLICATIONS

The aforementioned limitations aside, our findings provide ideas for pedagogy in introductory programming. Primarily, it is possible to categorize students based on PSS, to explore student constructivist learning improvements (Vygotsky, 1980; Ben-Ari, 2001). PSS levels may assist instructors to design constructivist-relevant assessments, to improve abstract reasoning skills for programming. Moreover, problem-solving skills are identified as one of the required "employability skills in the 21st century" (Suarta et al., 2017). That is, students should be able to succeed in studies and in the workplace. This requires assessment should to be (also) aligned with employment skills requirements. This means that while students' academic achievements may be highly valued, they may not suffice to secure employment, as employers expect students to have well developed problem-solving skills (Yorke, 2014). Therefore, assessing student PSS levels may help instructors to develop

instructional interventions and assessment tasks, to improve student academic self-efficacy, problem-solving and in learning programming.

Second, identifying effective approaches to teaching programing via application of valid methodological frameworks is important (Koulouri, Lauria, & Macredie, 2015). Specifically, PSS is a required skill to be able to understand the fundamentals of computing and should be learned while studying programming (Deek & McHugh, 2003). Therefore, we surmise that teaching problem-solving strategies before the course commences may improve novice students' conceptual knowledge. However, the difference between initial measures of student PSS at the beginning and at the end of the course should be measured to tune the adapted curriculum, pedagogy, and tools for supporting learning.

Third, integrating problem-based learning (PBL) with assessment tasks in the programming curriculum would enhance student self-efficacy and the sense of their own PSS. Notably, integrating problem-solving methodology and code development tasks would reduce the PSS gap between weak and good students. For example, PBL is a pedagogy that fosters positive development in student critical thinking. However, PBL implementation requires educators to come up with innovative and challenging tasks to realize its benefits (O'Grady & J, 2012). Therefore, attention should be paid to aligning formative and summative assessments improve skills transferability (Cain & Woodward, 2012; Morgan, et al., 2015). Cain et al. presented the constructive alignment portfolio model for teaching introductory programming (Cain & Woodward, 2012). This model was defined based on constructive learning theory, which advocates use of assessments that cover both conceptual knowledge and programming competencies. Other studies propose lesson plans associated with course specific learning outcomes, which would increase student learning and impact their performance in assessments aligned with standards and classroom instruction (Näsström & Henriksson, 2008; Lucas, Dippenaar, & Toit, 2014). Assessments should be defined in a learner-centred approach or as an active engagement to employ student in solving authentic problems, to support constructive alignment. It

involves a variety of activities including multiple drafts of written work, oral presentation by students, group projects, and service learning assignments; these activities facilitate student engagement. Specifically, alignment of learning outcomes, assessment, and instruction, may improve student performance. However, these activities require instructors to assign, evaluate and provide frequent and prompt feedback in order to help student to amalgamate their learning experiences, and to increase student-faculty contact (Webber & Tschepikow, 2013). Therefore, the challenge for educators is to develop diverse assessment methods for introductory courses, to reduce the summative assessment performance gap between students with good and poor PSS.

8. ACKNOWLEDGEMENTS

The authors wish to thank all members of ViLLE team research project group, Department of Future Technologies, University of Turku for their comments and support that greatly improved the manuscript. This research was supported fully by a University of Turku, Turku, Finland.

9. REFERENCES

Adachi, P. J., & Willoughby, T. (2013). More Than Just Fun and Games: The Longitudinal Relationships Between Strategic Video Games, Self-Reported Problem Solving Skills, and Academic Grades. *Journal of Youth and Adolescence, 42*(7), 1041-1052. doi:10.1007/s10964-013-9913-9

Akar, S. G., & Altun, A. (2017). Individual Differences in Learning Computer Programming: A Social Cognitive Approach. *Contemporary Educational Technology*, *8*(3), 195-213.

Akben, N. (2018). Effects of the Problem-Posing Approach on Students' Problem Solving Skills and Metacognitive Awareness in Science Education. *Research in Science Education*, 1-23. doi: https://doi.org/10.1007/s11165-018-9726-7

Argaw, A. S., Haile, B. B., Ayalew, B. T., & Kuma, S. G. (2017). The Effect of Problem Based Learning (PBL) Instruction on Students' Motivation and Problem Solving Skills of Physics. *Journal of Mathematics Science and Technology Education, 13*(3), 857-871. doi:10.12973/eurasia.2017.00647a

Askar, P. (2009). An investigation of factors related to self-efficacy for Java programming among engineering students. *The Turkish Online Journal of Educational Technology, 8*(1).

Bandura, A. (1977). Self-efficacy: Toward a Unifying Theory of Behavioral Change. *Psychological Review, 84*(2), 191-215.

Ben-Ari, M. (2001). Constructivism in Computer Science Education. *Journal of Computers in Mathematics and Science Teaching, 20*, 45-73.

Bester, L. (2014). *Investigating the problem-solving proficiency of second-year Quantitative Techniques students : the case of Walter Sisulu University.* PhD Dissertation, University of South Africa, Department of Education, Pretoria. Retrieved from http://hdl.handle.net/10500/14214

Bubica, N., & Boljat, I. (2014). Predictors of Novices programmer's performance. *Proceedings of ICERI2014 Conference*, (pp. 1536-1545). Seville, Spain.

Cain, A., & Woodward, C. J. (2012). Toward Constructive Alignment with Portfolio Assessment for Introductory Programming. *IEEE International Conference on Teaching, Assessment, and Learning for Engineering* (pp. H1B-11- H1B-17). Hong Kong: IEEE. doi:10.1109/TALE.2012.6360322

Chao, P.-Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*, 202-215. Retrieved from https://doi.org/10.1016/j.compedu.2016.01.010

Deek, F. P., & McHugh, J. A. (2003). Problem solving and cognitive foundations for program development: an integrated model. *Sixth International Conference on Computer Based Learning in Science (CBLIS)*, (pp. 266-271). Nicosia, Cyprus.

Dostál, J. (2015, February 12). Theory of problem solving. *Procedia - Social and Behavioral Sciences, 174*, 2798 – 2805. doi:10.1016/j.sbspro.2015.01.970

Erdogdu, F., & Erdogdu, E. (2015). The impact of access to ICT, student background and school/home environment on academic success of students in Turkey: An international comparative analysis. *Computers & Education, 82*, 26-49. Retrieved from https://doi.org/10.1016/j.compedu.2014.10.023

Eskin, M. (2013). Components of Problem Solving. In M. Eskin, *Problem Solving Therapy in the Clinical Practice* (pp. 21-27). Elsevier.

Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. *Journal of Computer Assisted Learning, 32*(6), 576–593. doi:10.1111/jcal.12155

Ghasemi, A., & Zahediasl, S. (2012). Normality Tests for Statistical Analysis: A Guide for Non-Statisticians. *International Journal of Endocrinology Metabolism, 10*(2), 486-489.

H.Nowaczyk, R. (1984). The relationship of problem-solving ability and course performance among novice programmers. *International Journal of Man-Machine Studies, 21*(2), 149-160. Retrieved from https://doi.org/10.1016/S0020-7373(84)80064-4

Heppner, P. P. (1982). The Problem Solving Inventory. New York: The American Psychological Association.

Heppner, P. P., & Krauskopf, C. J. (1987). An Information-Processing Approach to Personal Problem Solving. *The Counseling Psychologist, 15*(3), 371-447. doi:https://doi.org/10.1177/0011000087153001

Heppner, P. P., & Petersen, C. H. (1982). The Development and Implications of a Personal Problem-Solving Inventory. *Journal of Counseling Psychology, 29*(1), 66-75.

Jauke, J., & Kossowski, T. (2011). Comparison of values of Pearson's and Spearman's Correlation Coefficients on the same sets of Data. (A. Kostrzewski, Ed.) *Quaestiones geographicae, 30*(2), 87-93.

Jolliffe, F. R. (1990). Assessment of the Understanding of Statistical Concepts. *Third International Conference on Teaching Statistics. 1*, pp. 461-466. Otago University Press.

Kappelman, L., C.Jones, M., Johnson, V., R.Mclean, E., & Bonnme, K. (2016). Skills for success at different stages of an IT professional's career. *Communications of the ACM, 59*(8), pp. 64-70. doi:10.1145/2888391

Koulouri, T., Lauria, S., & Macredie, R. D. (2015). Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches. (C. Hundhausen, Ed.) *ACM Transactions on Computing Education, 14*(4), 26.1-26.27. doi:10.1145/2662412

Laakso, M.-J., Kaila, E., & Rajala, T. (2018). ViLLE – collaborative education tool: Designing and utilizing an exercise-based learning environment. *Education and Information Technologies,* 1655-1676. Retrieved from https://doi.org/10.1007/s1063017-9659-1

Lee, M. O., & Thompson, A. (1997). Guided Instruction in Logo Programming and the Development of Cognitive Monitoring Strategies among College Students. *Journal of Educational Computing Research, 16*(2), 125-144. doi:https://doi.org/10.2190/PW3F-HLFD-1NNJ-H77Q

Lishinski, A., Yadav, A., Enbody, R., & Good, J. (2016). The Influence of Problem Solving Abilities on Students' Performance on Different Assessment Tasks in Introductory Programming. *SIGCSE '16* (pp. 329-324). Memphis, TN, USA: ACM. doi:10.1145/2839509.2844596

Lishinski, A., Yadav, A., Good, J., & Enbody, R. (2018). Learning to Program: Gender Differences and Interactive Effects of Students' Motivation, Goals, and Self-Efficacy on Performance. *ICER '16 Proceedings of the 2016 ACM Conference on International Computing Education Research* (pp. 211-220). Melbourne, VIC, Australia: ACM. doi:10.1145/2960310.2960329

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Morten Lindholm, R. M., . . . Thomas, L. (2004). A Multi-National Study of Reading and Tracing Skills in Novice Programmers. *SIGCSE Bulletin, 36*(4), 119-150.

Longi, K. (2016). *Exploring factors that affect performance on introductory programming courses.* Master's Thesis, University of Helsinki, Department of Computer Science, Helsinki.

Lucas, K., Dippenaar, S., & Toit, P. D. (2014). Analysis of assessment practice and subsequent performance of third year level students in natural sciences. *Africa Education Review*, 563-583. Retrieved from https://doi.org/10.1080/18146627.2014.935004

Martin, R. G. (1971). Plagiarism and Originality: Some Remedies. *The English Journal, 60*(5), 621-625. Retrieved from http://www.jstor.org/stable/813078

Mayer, R. E. (1998). Cognitive, metacognitive, and motivational aspects of problem solving. *Instructional Science, 26*(1), 49-63. Retrieved from https://doi.org/10.1023/A:1003088013286

Md.Yunus, A. S., Hamzah, R., Tarmizi, R. A., Abu, R., Md.Nor, S., Ismail, H., . . . Bakar, K. A. (2006). Problem Solving Abilities of Malaysian University Students. *International Journal of Teaching and Learning in Higher Education, 17*(2), 86-96.

Morgan, M., Sheard, J., Butler, M., Falkner, K., Simon, & Weerasinghe, A. (2015). Teaching in First-Year ICT Education in Australia: Research and Practice. In D. D'Souza, & K. Falkner (Ed.), *17th*

Australasian Computing Education. 160, pp. 27-30. Sydney, Australia: Australian Computer Society.

Näsström, G., & Henriksson, W. (2008). Alignment of standards and assessment: A theoretical and empirical study of methods for alignment. *Electronic Journal of Educational Psychology, 6*(3), 667-690.

O'Grady, & J, M. (2012). Practical Problem-Based Learning in Computing Education. *ACM Transactions on Computing Education, 2*(3), A1-A14.

Omiwale, J. B. (2011). Relationship Between Problem-Solving Ability and Achievement in Physics Among SeniorSecondary School Students in Osun State, Nigeria. (D. Adewuyi, Ed.) *The African Symposium: An online journal of the African Educational Research Network, 11*(1), 158-165.

Ozus, E., Celikoz, M., Tufan, M., & Erden, F. (2015). Interpersonal Problem Solving Abilities of Students of Professional Education Faculty Dressing Programme of Selcuk University. *4th WORLD CONFERENCE ON EDUCATIONAL TECHNOLOGY RESEARCHES, WCETR2014* (pp. 456 – 462). Elsevier. doi:10.1016/j.sbspro.2015.04.827

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*(2), 137-168. Retrieved from https://doi.org/10.1016/0732-118X(84)90018-7

Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science, 45*, 583–602. doi:10.1007/s11251-017-9421-5

Rajala, T., & Erkki Kaila, M.-J. L. (n.d.). *ViLLE*. (University of Turku) Retrieved 10 20, 2015, from http://villeteam.fi/index.php/en/

Rosenfeld, P., Booth-Kewley, S., & Edwards, J. E. (1996). Responses on computer surveys: Impression management, social desirability, and the big brother syndrome. *Computers in Human Behavior*, 263-274.

Sabin, M., Alrumaih, H., Impagliazzo, J., Lunt, B., & Zhang, M. (2017). *Information Technology Curricula 2017: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology*. New York, NY, USA: ACM and IEEE. doi:10.1145/3173161

Sharma, R., & Shen, H. (2018). Does Education Culture Influence Factors in Learning Programming: A Comparative Study between Two Universities across Continents. *International Journal of Learning, Teaching and Educational Research, 17*(2), 1-24. Retrieved from https://doi.org/10.26803/ijlter.17.2.1

Shrout, J. R., & Witty, T. E. (1990). Problem-Solving Appraisal, Self-Reported Study Habits, and Performance of Academically At-Risk College Students. *Journal of Counseling Psychology, 37*(2), 203-207. doi:10.1037//0022-0167.37.2.203

Suarta, I. M., Suwintana, I. K., Sudhana, I. G., & Hariyanti, N. K. (2017). Employability Skills Required by the 21st Century Workplace: A Literature Review of Labor Market Demand. *1st International Conference on Technology and Vocational Teachers (ICTVT 2017). 102*, pp. 337-342. Atlantis Press. Retrieved from http://creativecommons.org/licenses/by-nc/4.0/

Trumbull, E., & Lash, A. (2013). Understanding Formative Assessment: Insights from Learning Theory and Measurement Theory. San Francisco: WestEd.

Tu, J.-J., & Johnson, J. R. (1990, June). Can computer programming improve problem-solving ability? *ACM SIGCSE Bulletin, 22*(2), pp. 30-33. doi:10.1145/126445.126451

Uysal, M. P. (2014). Improving First Computer Programming Experiences: The Case of Adapting a Web-Supported and Well- Structured problem-Solving Method to a Traditional Course. *Contemporary Educational Technology, 5*(3), 198-217.

Webber, K. L., & Tschepikow, K. (2013). The role of learner-centred assessment in postsecondary organisational change. *Assessment in Education: Principles, Policy & Practice, 20*(2), 187-204. Retrieved from https://doi.org/10.1080/0969594X.2012.717064

Veerasamy, A. K., & Souza-Daw, T. d. (2012). Impact of ICT on Society - Higher Education students in South-East Asia. *IEEE Symposium on Business, Engineering and Industrial Applications* (pp. 275-278). Bandung: IEEE.

Veerasamy, A. K., Daryl D'Souza, R. L., & Laakso, M.-J. (2018). The impact of prior programming knowledge on lecture attendance and final exam. *Journal of Educational Computing Research, 0*(0), 226-253. doi:10.1177/0735633117707695

Veerasamy, A. K., D'Souza, D., Lindén, R., Kaila, E., Laakso, M.-J., & Salakoski, T. (2016). The Impact of Lecture Attendance on Exams for Novice Programming Students. *International Journal of Modern Education and Computer Science (IJMECS), 8*(5), 1-11. doi:10.5815/ijmecs.2016.05.01

White, H. B., Benore, M. A., Sumter, T. F., Caldwell, B. D., & Bell, E. (2013, September 10). What skills should students of undergraduate biochemistry and molecular biology programs have upon graduation? *Biochemistry and Molecular Biology Education, 41*(5), 297-301. doi:10.1002/bmb.20729

Voskoglou, M. G., & Buckley, S. (2012). Problem Solving and Computers in a Learning Environment. *Egyptian Computer Science Journal, 36*(4), 28-46.

Vygotsky, L. S. (1980). *Mind in society: The development of higher psychological processes.* Harvard university press.

Yorke, M. (2014). Employability in higher education: what it is – what it is not. *Learning & Employability-Series one*. Learning and Teaching Support Network (LTSN) and the Enhancing Student Employability Co-ordination Team (ESECT).

Yukselturk, E., & Altiok, S. (2017). An investigation of the effects of programming with Scratch on the preservice IT teachers' self-efficacy perceptions and attitudes towards computer programming. *British Journal of Educational Technology*, 789-801. doi:10.1111/bjet.12453

Yurdugül, H., & Aşkar, P. (2013). Learning Programming, Problem Solving and Gender: A Longitudinal Study. *Procedia - Social and Behavioral Sciences. 83*, pp. 605-610. ELSEVIER. Retrieved from https://doi.org/10.1016/j.sbspro.2013.06.115

Özen, Y. (2016). Can I Solve the Problem? A Program Trail on Problem Solving Skill. *American Journal of Applied Psychology, 4*(1), 1-10. doi:DOI:10.12691/ajap-4-1-1

Özyurt, Ö. (2015). Examining the Critical Thinking Dispositions and the Problem Solving Skills of Computer Engineering Students. *Eurasia Journal of Mathematics, 11*(2).

Table 1. Meanings of the problem-solving skills subscale.

Scales of PSI	Meaning	Interpretation of the PSI possible scores	
		Low score*	High score**
Problem-solving confidence (PSC)	This scale reflects a basic belief and trust in one's PSS to effectively cope with problems.	11	66
Approach-avoidance style (AAS)	This scale reflects a general tendency to approach or avoid problem-solving activities.	16	96
Personal control (PC)	This scale reflects peoples' control of their emotions while problem solving	5	30
Total		32	192

* Low scores indicates that student perceives himself/herself as effective problems solver

** High scores indicates that student perceives himself/herself as poor problem solver

Table 2. Students by perceived problem-solving skills level declared in the course entry survey

PSS score band	PSS level	Meaning	Number of students
32-75	1 (High)	Effective problem solver	26
76 -100	2 (Mid)	Moderate problem solver	93
101 - 192	3 (Low)	Poor problem solver	47
		Total	166

Table 3. Spearman's rank correlation coefficient results: PSS, HE, DE, and FE

Variables		**Spearman's Rank correlation – year**	**Sig (2-**
No. of students : 166		**2016**	**tailed)**
(i)	PSS and HE	-165*	.034
(ii)	PSS and DE	-199*	.010
(iii)	PSS and FE	-254**	.001

*. Correlation is significant at the 0.05 level (2-tailed).

**. Correlation is significant at the 0.01 level (2-tailed).

Table 4. The relationship between PSS (PSS levels) and student HE and DE mean scores

PSS_Level	No. of students	HE_Mean	DE_Mean	HE and FE Mean
Level 1	26	93.74335	70.86538	82.304365
Level 2	93	89.88653	68.33333	79.10993
Level 3	47	88.29297	62.39362	75.343295
Overall	166	90.03942	67.04819	

Table 5. The relationship between PSS (PSS levels) and student FE mean score

PSS_Level	No. of students	FE-Mean	95% confidence interval for mean Lower	Upper
Level 1	26	82.90598	73.1170	92.69496
Level 2	93	78.88889	74.4801	83.29764
Level 3	47	68.46336	61.6883	75.23840

Table 6. Multiple linear regression results for PSS on FE scores

Model summary and coefficients	R	Adjusted R^2	Remarks
H5: Results of PSS on FE scores	0.222	0.043	The coefficient for PSS (0.222) is significantly different from 0 because its p-value is 0.004, which is smaller than 0.050. So, the regression model is a good fit of the data.

Figure legends:

Figure 1. A screenshot of the automatic feedback generated by ViLLE

Figure 2. A sample screen shot of coding exercises for a DE session

Figure 3. The relationship between PSS (PSS levels) and student HE and DE mean scores

Figure 4. The relationship between PSS and mean FE scores

Figure 5. The impacts of PSS on student FE scores – Mark ranges / grade points

Figure 6. Frequency graph for student final exam grade