Engineering students' strategies to learn programming correlate with motivation and gender

1st Maria Weurlander
Department of Education, Stockholm University
Stockholm, Sweden
maria.weurlander@edu.su.se
orcid.org/0000-0002-3027-514X

2nd Kristina von Hausswolff
Department of Information Technology
Uppsala University, Uppsala, Sweden
kristina.von.hausswolff@it.uu.se
orcid.org/0000-0003-1750-8647

***Abstract*—This full research paper reports on a study where we investigated engineering students' learning to program. In particular, we explored differences in students' strategies, how they work in the computer lab, and if this is related to their motivation, gender and overall tendency to engage in thinking. We gathered survey data from first year engineering students in an introductory, compulsory programming course. The survey consisted of established instruments and items constructed by us. 67 students answered the survey (response rate of 43%). 18% of the students that answered the survey did not have previous experiences of programming, and 43% were female. An exploratory factor analysis of the items relating to how students learn programming revealed three factors: 1) the individual thinker, 2) the social reader, and 3) the interactive problem-solver. We found gender differences relating to the second factor; female students reported more frequent use of the social reader strategy. There were also differences in reported used strategies and how students worked in the lab, previous experiences, need for cognition and motivation. These differences indicate that the factors individual thinker and interactive problem-solver were privileged in this programming course. Further research is needed to explore these findings in different educational contexts.**

***Index Terms*—novice programmers, introductory programming, learning strategies, pragmatism, gender differences, motivation**

I. Introduction

It is well known that students find it difficult to learn programming and the dropout rates from programming courses are high [1]. There are many reasons as to why students find programming difficult and may consider dropping out. Lack of motivation is one important reason [2], also, drop-out students found learning programming time consuming and involving hard work, and that falling behind in their work influenced their decision to leave [3]. Another issue for concern is the under-representation of women in computer science (CS) [4]. Moreover, working hands-on at the computer seems to be an important aspect of learning to program [5], [6]. Students learn both theoretical concepts and practical skills when interacting with the computer. The learning process seems to go back and forth involving both thinking and doing [7], [8]. Furthermore, previous research has found that students learning to program use various strategies to get 'unstuck' [9]. It also seems that successful CS students spend more time working hands-on at the computer [6].

Taken together, the research summarized above reports on that learning programming is demanding. However, in order to better understand why learning programming is difficult for many students, we need to investigate specifically how students differ when learning programming. This led us to investigate in more detail what strategies are used when learning to program. In particular, we aimed to explore if there are differences in students' strategies, how they work in the computer lab, and if this is related to their overall tendency to engage in thinking and their motivation. We also wanted to explore if there are gender differences relating to learning strategies.

The current study is part of a larger research project aiming to explore how novices learn to program, involving investigations in both experimental set-ups and authentic educational settings [5], [7], [10], [11]. The context of the present study was an introductory programming course, or module, for first year engineering students at a Swedish university. The course was taught part-time in parallel to other courses and was compulsory for the students. The students were non-CS majors but could, if they wanted to, choose to continue into a CS major trajectory during their second year of study.

II. Background

A. Learning to program

Learning programming is demanding and requires various cognitive processes, including cognitive skills such as recognition, evaluation, abstraction, and problem-solving [12]. However, learning programming also involves *doing*, i.e. writing code hands-on [6], [13]. The thinking and doing seems to interplay so that students think and do (write code, try ideas) intertwined [7]. There is agreement that learning hands-on at the computer is essential to students' learning programming, and computer labs and programming assignments are common activities in CS courses [14], [15]. Successful students have been found to spend considerably more time working hands-on at the computer compared to less successful students [6]. Students often work on programming assignments together in small groups or pairs, where one of the students has control over the keyboard and the others contribute to solving the task by being involved in discussions or providing help. Working together with a programming assignment can be beneficial for

students' learning [16]. However, social dimensions come into play which can affect the collaboration between the students, and in turn how they solve the task. Working together with another student equally engaged in solving the assignment is more effective and supportive [10], [17]. Moreover, how well students know their partner may influence how they allow the other students to work hands-on and take turns at the keyboard, and consequently what they learn [5], [18]. When students have notable differences in programming knowledge and skills, the teamwork seems to be more unequal, and less experienced students may risk not to learn from the collaboration [10], [19].

The fact that women are under-represented in CS, despite efforts to attract more women, is discouraging. Numerous studies have explored why this may be the case [15]. Beyer [4] found that men had significantly higher computing self-efficacy than women, and that women rated their CS ability lower than men. Moreover, women felt that intrinsic factors were important in choosing a career, and were less likely to take another CS course in the future compared to men. These differences were salient, despite no significant differences i CS grades. Women have also been found to have lower sense of belonging to computing compared to men, and this feeling decreased further during an introductory course [20].

B. Learning strategies and motivation

There is a substantial body of research that reports on how students differ in their approach to learning, their conceptions of knowledge and learning, their level of self-regulation and their motivation to study [21]–[23]. Students tend to seek meaning and understanding, or try to learn enough to pass the exam. Their intentions correlate with their study strategies and also what they learn [24]. Researchers have described these as students' overall learning pattern, emphasizing the importance of the context and that students' learning patterns is "a result of the interplay between personal and contextual influences" [22, p. 271].

In addition to the approach or learning pattern students adopt, their motivation significantly influence their learning. Various factors contributing to both extrinsic or intrinsic motivation play important roles in learning programming [25]. Intrinsic motivation means that a student is driven by a genuine interest and personal satisfaction, rather than external pressure or seeking a reward (i.e. grade). The latter is referred to as external motivation. A study of first year IT students found that female students were more motivated than male students, and scored significantly higher on both intrinsic and extrinsic motivation [26]. Also, intrinsic motivation was correlated to higher academic achievement whereas extrinsic motivation correlated to lower academic achievement. Another study reports that students prior programming experience is related to their motivation; students with no prior knowledge were more driven by external motivation compared to students with prior knowledge [27].

III. Theory

A. Pragmatism as an theoretical foundation

Our theoretical standpoint in this study is a pragmatic view of learning. Dewey, a pragmatic thinker, describes the learning process in terms of *experience*, *inquiry* and *continuity* [28, p. 58]. In learning, action and thinking are intertwined according to Dewey, and knowledge is obtained through *experiencing* consequences of actions. Through thinking, symbols and operations are created as a way of simulating different consequences of an action without experiencing them. This allows us to better handle changes in the environment [29], which is how knowledge primarily reveals itself [28]. The individual seeks a balance with the environment and when we face situations that are new to us an opportunity for learning opens up. This process is called an *inquiry* by Dewey [28, p. 58]. Through reflection and available resources, the individual formulates hypotheses about different ways of acting and possible consequences. In a Deweyan framing, this would be a sequence of inquiries linked together, forming understanding in the end. The individual uses prior *experiences* as a starting point in the process of *inquiry* according to Dewey's principle of *continuity*. This principle also includes direction in the form of purpose in education.

We have used this pragmatic view of knowledge and learning in previous studies about programming [7], [10], [30], and it is also used by researchers in other educational disciplines in order to understand educational practices (e.g. [31], [32]). Interacting with the computer environment is an essential part of learning to program, described by the second author as *practical thinking* while programming [7]. *Practical thinking* builds upon the Deweyan rejection of the dichotomy of practice and theory. Dewey states: "The concrete fact behind the current separation of body and mind, practice and theory, actualities and ideals, is precisely this separation of habit and thought. Thought which does not exist within ordinary habits of action lacks means of execution." [33, p. 17].

B. Need for cognition and motivation

In addition to the overall theoretical perspective of pragmatism described above, two other theoretical concepts are important for learning and achievement, and therefore also to the present study: *need for cognition* and *motivation*. A brief description of these concepts will be presented here. Learning programming is cognitively demanding and requires both lower and higher cognitive processes as well as various cognitive skills such as recognition, evaluation, abstraction, and problem-solving [12]. Humans differ in their tendency to enjoy thinking and engage in cognitively demanding tasks. Cacioppo and Petty [34] developed a scale to measure this tendency, called *Need for Cognition* (NFC). Students with a high score on NFC are more likely to engage in and put more effort into challenging tasks compared to students with lower scores. The tendency to enjoy and engage in thinking have been shown to relate to university students' academic achievement [35]. Similar findings have been found in secondary students learning to program [5].

Motivation is the drive or desire to accomplish something; it can be to learn a new language, or pass an assessment task. Motivation is thus directed towards an end, a goal, and relates to affect, effort and action. Moreover, motivation is a multidimensional phenomenon, including different types of motivation based on the different reasons behind the action [36]. Motivation is often divided into "*intrinsic motivation*, which refers to doing something because it is inherently interesting or enjoyable, and *extrinsic motivation*, which refers to doing something because it leads to a separable outcome" [36, p. 55]. The *Intrinsic Motivation Inventory* (IMI) [37] has been developed as a tool to measure subjects' levels of intrinsic motivation, including items relating to interest, enjoyment, perceived competence, effort, importance, and pressure related to a certain situation.

Following our theoretical stance of pragmatism, action (doings) and thinking are intertwined when learning to program (*practical thinking*). The two inventories described above were chosen to capture aspects of importance to students learning programming relating to thinking and intrinsic motivational outcomes respectively. Aspects of importance relating to *doing* were captured by items constructed by us, as described below.

IV. Method and research design

A. Research design and Data collection

The present study used a cross-sectional study design where we collected survey data from first year engineering students in an introductory programming course in 2018. Computer science (CS) was not the students' major subject, but they could specialize in CS later during their education. The course ran part time over one semester, in parallel to other courses. The survey was distributed manually to students after about 6-7 weeks, in connection with the mid-course exam, and the students answered on paper. In order to capture how students learn programming (doing aspects), we constructed 13 items (see table IV-C1) related to how they learn programming and three items of how they worked in the computer lab (by themselves, in pairs at one computer or together but on separate computers). We also constructed 21 items that related to their experiences of learning programming more broadly, i.e. emotional aspects, interaction with the programming environment and aspects relating to working with others. The items was constructed based on observations and interviews, conducted by the second author as part of the larger research project, with students taking the course. The findings from observations and interviews have been, or will be, published elsewhere [7], [10]. The survey also included the following background information and items from established instruments:

- Background questions on gender and prior experiences in programming.
- *Need For Cognition* (NFC) inventory [38], an instrument (16 items) that measures the tendency for an individual to engage in and enjoy thinking. We used a Swedish version. The instrument is designed to measure attitudes relatively stable over time.
- *Intrinsic Motivation Inventory* (IMI) [37], which consists of four subscales: interest-enjoyment (7 items), perceived competence (6 items), effort-importance (4 items), and tension-pressure (4 items). We used a Swedish version.

67 out of ca 155 students answered the survey (response rate of 43%), 55% of the students that answered were male and 43% female. One student did not state his/her gender and was excluded from the gender category due to the small size. 81% of the students that answered the survey did not have previous experiences of programming.

B. Data pre-processing of the variables

The survey was distributed and answered on paper. The data was afterwards entered to Excel and exported to SPSS 26, where all the statistical analyses were conducted. Some data pre-processing was done before the statistical analyses were conducted, which is described in detail below.

- NFC was calculated with values from participants that answered more than 50 % of the NFC scale (6 % excluded). The values were an average of the questions answered raging from 4 to -4, with 4 considered a high NFC score (high score of enjoyment of thinking/problem-solving).
- All of the IMI subscales were calculated, separately, with values from participants that answered more than 50 % of the questions per subscale (10 % excluded) and the values were an average of the questions answered, ranging from 1 to 7, with 7 considered the highest value.
- The three factors that emerged during the factor analysis were calculated during that process and all the factors ranged from a value of -2.5 to a value of 2.5. where 2.5 were the highest value on that factor.
- The index "continue with programming" was calculated by taking the maximum of the two items "I intend to continue programming because it gives me pleasure" and "I intend to continue programming because I'm good at it". Both values were measured on a likert scale (1-7) (1 = do not agree at all, 7 = completely agree).
- The variables measuring how students worked in the lab sessions used in the cluster analysis were measured on a scale 1 to 4, (1=rare, 4=very often), see ref IV-C2.
- All the other items included in this survey were measured on a likert scale (1-7) (1 = do not agree at all, 7 = completely agree).

To be able to examine if there were any significant differences between the groups, all the variables were tested for normal distribution with the Kolmogorov-Smirnov test. This test was chosen because of the size of the sample ($50 < n < 100$). The variables NFC, IMI (interest-enjoyment) , IMI (perceived competence), IMI (tension-pressure), Social reader (f), Interactive problem-solver (f) each had a non-significant result on the Kolmogorov-Smirnov test and are therefore considered to have approximately a normal distribution. Consequently, a parametric test was used on those variables (eg. independent t-test). The variables IMI (effort-importance), Individual thinker

TABLE I
ITEMS ABOUT HOW I LEARN PROGRAMMING (LIKERT SCALE 1-7)

No	*Items*
1	I learn well by discussing with my lab partner
2	I learn well when I test an idea by writing code
3	I learn well by asking the TA
4	I learn well by reading lecture notes
5	I learn well by googling the answers for my questions
6	I learn well by reading example code
7	I learn well when I write code by myself
8	I learn well when I sit next to someone who writes code
9	I learn well when I draw sketches of how the program works
10	When I get stuck, I prefer to ask others first
11	When I get stuck, I first try to figure out it out my self
12	I think out the solution first and then I implement it
13	I write code and think out the solution at the same time

TABLE II
PATTERN MATRIX - LEARNING STRATEGIES

No	*Individual thinker*	*Social reader*	*Interactive problem-solver*
2	.71		
3		.67	
4		.69	
5			.77
7	.81		
8		.76	
10		.85	
11	.60	$-.44$	
12	.76		$-.32$
13			.77

Extraction Method: Principal Component Analysis.
Rotation Method: Oblimin with Kaiser Normalization.
Loading factors less than .3 are excluded for readability reasons.

(f) and Continuing with programming (i) had a significant result on the Kolmogorov-Smirnov test, and therefore we used a non-parametric tests on those variables (eg. independent Mann-Whitney test).

C. Data analysis

1) Exploratory factor analysis: As a first step, we conducted an Exploratory factor analysis (EFA) on the 13 variables on resources used to learn programming described in table I. We found three factors that captured different strategies when learning programming. Of the initial 13 items, three items were excluded (no 1, 6 and 9) because they did not fit the model and loaded on different factors. Item 6, about using code examples, could be interpreted in two ways: one could read the example code from a teacher or from the internet. We believe this ambiguity could explain why this item did not have a clear loading on a single factor. Also, we mean that the item regarding reading lecture notes (no 4) captured the interpretation of reading example code from a teacher, and the item about googling (no 5) captured the interpretation of reading example code from internet. Thus, item no 6 was excluded. The second item that was excluded was no 1, discussing with my lab partner. Other questions in this survey revealed that most of the students did not work with a lab partner, making the item irrelevant to most of the students. This may explain why the item did not load clearly on any of the factors.

The third item we excluded was no. 9, about sketching when solving problems in programming. The analysis revealed that this item was not related to the others and only confused the analysis. It was therefore excluded. To make sure that this exclusion does not bias the result of the factor analysis we tested an EFA on a model with no. 9 included. This test resulted in the same factors and the same items loading on them, with only minor differences in loading factor values. The biggest change on the loading factors was an increase of .07 on item no. 7, followed by a .06 increase of no. 11, and a decrease of .06 of no. 2. From this test we conclude that it is safe to exclude no. 9.

The exploratory factor analysis of the remaining 10 items relating to how students learn programming revealed three distinct factors. This model explained 65% of the variance. The three factors are: 1) *Individual thinker*, 2) *Social reader*, and 3) *Interactive problem-solver* (see table II). These factors correspond to three different strategies students used to learn programming.

2) Cluster analysis: The second step in the analysis involved cluster analysis. The students could choose how to work at the lab sessions: alone, together in pairs with two computers, or together in pairs working on one computer. The students reported if they worked alone or in pairs during the labs that were mandatory, responding to three statements on the survey. The cluster analysis were calculated using two of the items about how the student worked during the lab sessions and if the students stated that they had prior experience of programming before the course. The two items were in answer to this question: "How did you mostly work during the labs? Take a stand on each statement and circle the option that best suits you. 1 = rare, 4 = very often." The first statement being: " I worked on a computer by myself", and the second statement being "I was working with a lab partner on one computer". The third statement that did not contribute to the cluster analysis and therefore where omitted where answer to the same questions as the other two: "I worked with a lab partner but we each wrote on our separate computers". The cluster analysis revealed that the students with prior experiences of programming always chose to work alone, as did half of the students without prior experiences. The remaining students without prior experiences worked mostly together in pairs, sometimes on one computer and sometimes with two computers. We identified these groups with a cluster analysis with a good fit (average silhouette = 0.6). The three groups being: 1) non-prior experiences and worked in pairs ($n = 22$), 2) non-prior experiences and worked alone ($n = 25$), and 3) prior experiences and worked alone ($n = 12$) a total of 59 students of the 67 students due to missing data.

V. Ethical considerations

Ethical guidelines concerning research involving human beings were strictly followed. The students were informed orally about the study prior to data collection. Participating in the study, i.e. answering the survey, was voluntary. The integrity of the participants was protected since the data was anonymous, no personal or sensitive information collected and the results only presented on a group level.

VI. Results

A. Group comparison

In this section we compare groups found within our student survey data, and report on differences between those groups. The first group comparison is based on how the students chose to work in the lab sessions, mostly together with a lab partner versus mostly alone. The grouping is a result of our cluster analysis, described in section IV-C2. The group of students working alone correspond to groups 2-3 from the cluster analysis, while students working together correspond to cluster group 1. Secondly, comparing the group with prior experience of programming with the group without. All of the students with prior experiences of programming also chose to mostly work alone. Finally, we also compared differences between male and female students. Only significant results are reported, see Table III for an overview.

1) Comparisons between students who worked alone with students that worked together with a lab partner: Students who mostly worked together during the lab sessions scored significantly lower on the NFC ($M = 1.26$, $SD = .84$) than students that mostly worked alone ($M = 1.88$, $SD = .84$), $t(57) = -2.76$, $p = .008$, $d = .74$. They also scored lower on the IMI interest-enjoyment ($M = 4.26$, $SD = .75$) and the perceived competence ($M = 3.28$, $SD = .91$) subscales compared to students who mostly worked alone ($M = 4.78$, $SD = 1.21$ and $M = 4.00$, $SD = 1.60$), $t(57) = -2.01$, $p = .050$, $d = .51$ and $t(57) = -2.19$, $p = .033$, $d = .51$ respectively. Levene's test indicated unequal variances ($F = 4.04$, $p = .049$ interest-enjoyment and ($F = 6.24$, $p = .015$) perceived competence), so degrees of freedom were adjusted from 57 to 56.8 for both subscales but shown as the nearest whole number above. Furthermore, students who worked together with a lab partner scored lower on the IMI effort-importance subscale ($Mdn = 5$) than students who worked alone ($Mdn = 6$), $U = 582$, $p = .006$, $r = .36$. We also found that the factor *Individual thinker* was significantly lower for students who mostly worked together at lab sessions ($Mdn = -.23$) than for students that mostly worked alone ($Mdn = .44$), $U = 576$, $p = .004$, $r = .38$. Also, the index "Continuing with programming" was significantly lower for the students that mostly worked together at lab sessions ($Mdn = 4$) than for students that mostly worked alone ($Mdn = 5$), $U = 500$, $p = .022$, $r = .31$.

The results indicate that students who chose to work together during lab sessions tended to enjoy thinking to a lesser degree, be less interested in programming, felt that they were less competent and valued programming less compared to students who mostly worked individually. It is therefore not surprising that the group of students who worked together also wanted to continue with programming to a lesser degree.

TABLE III
Significant differences between groups, * 0.05, **0.01

Factors	***Lab work (A/T)***	***Experience (Yes/No)***	***Gender (M/F)***
NFC_1	Alone**	Experience*	
IMI (interest-enjoyment)$_1$	Alone*	Experience*	
IMI (perceived competence)$_1$	Alone*	Experience**	
IMI (tension-pressure)$_1$		No experience*	
IMI (effort-importance)$_2$	Alone**	Experience*	
Individual thinker (f)$_2$	Alone**	Experience**	
Social reader(f)$_1$			F**
Interactive problem-solver(f)$_1$		Experience**	
Continue programming (i)$_2$	Alone*	Experience*	M*

$_1$: Independent t-test, $_2$: Mann-Whitney test
Column 1: Compare mean/rank between the group of students that mostly worked individually at lab sessions with the students that mostly worked together with a lab partner (Grouping: alone/together)
Column 2: Compare mean/rank between the group of students that have no no prior experience of programming before the course with the students that have programming experience (Grouping: Experience/No experience)
Column 3: Compare mean/rank between the male and female students (Grouping: Male/Female).
Only significant results are shown. When significant differences are present the group with the highest value on that factor are shown.

2) Comparisons between students with no prior experience of programming with students that have programming experience: Students with no prior programming experience scored significantly lower on the NFC test ($M = 1.45$, $SD = .89$) than students with prior experience ($M = 2.15$, $SD = .83$), $t(61) = -2.45$, $p = .017$, $d = .81$. Students with no prior experience also scored lower on the IMI interest-enjoyment ($M = 4.39$, $SD = 1.16$), and the perceived competence ($M = 3.29$, $SD = 1.21$) subscales compared to students with prior experience ($M = 5.18$, $SD = .64$) and ($M = 5.32$, $SD = 1.04$), $t(58) = -2.27$, $p = .027$, $d = .85$. and $t(58) = -5.33$, $p < .001$, $d = 1.80$ respectively. Furthermore, students with prior experience scored higher on the IMI effort-importance subscale ($Mdn = 6$) compared to students with no prior experience of programming ($Mdn = 5.5$), $U = 410$, $p = .024$, $r = .29$. However, the IMI tension-pressure subscale score was significantly higher for students with no prior experience of programming ($M = 3.95$, $SD = 1.36$) than students with prior experience ($M = 2.85$, $SD = 1.63$), $t(58) = 2.40$, $p = .019$, $d = -.73$. The analyses indicated that the factor *Individual thinker* was significantly lower for students with no prior experience of programming ($Mdn = .11$) than for students with prior experience ($Mdn = .84$), $U = 432$, $p = .008$, $r = .33$. The same was found for the factor *Interactive problem-solver*: students with no prior experience of programming ($M = -.20$, $SD = .90$) compared to students with prior experience ($M = .93$, $SD = .90$), $t(61) = -3.72$, $p < .01$, $d = 1.25$. Students without prior experience scored significantly lower on "Continuing with

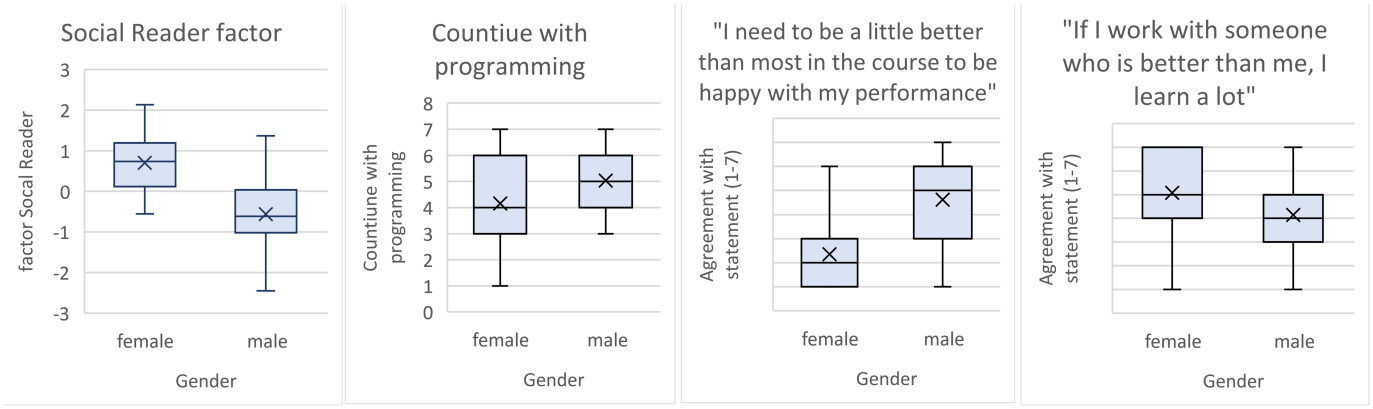

Fig. 1. The graph displays significant differences between female and male in our study regarding the learning strategic *Social reader*, the index "Continue with programming, agreement with the statement: "I need to be a little better than most in the course to be happy with my performance" and agreement with the statement: "If I work with someone who is better than me, I learn a lot"

programming" ($Mdn = 4$) compared to students with prior experience ($Mdn = 6$), $U = 367$, $p = .037$, $r = .28$.

The results suggest that students with prior experiences of programming (12 students) used the *Individual thinker* strategy and the *Interactive problem-solver* strategy more than students without prior experience. This group also scored higher on the NFC test and felt more motivated and less pressured. They also felt competent and plan to continue with programming after the course because felt they were good at programming.

3) Comparisons between male and female students: Female students scored significantly higher on the factor *Social reader* ($M = .70$, $SD = .70$) than male students ($M = -.57$, $SD = .85$), $t(60) = -6.30$, $p < 0.01$, $d = 1.62$ (independent t-test). Furthermore, the index "Continuing with programming" was significantly lower for female students ($Mdn = 4$) than for male students ($Mdn = 6$), $U = 263$, $p < .049$, $r = .27$ (Mann-Whitney test).

The female students in our study seemed to prefer the *Social reader* strategy when learning to program, and were also to a lesser degree than male students considering continuing with programming. These results led us to explore our data further in order to shed more light on the gender differences. In the end of our survey we asked the students to agree, on a likert scale (1-7), with items relating to programming more broadly (21 items in total). These items were based on interviews conducted previously and reported elsewhere [7], [10]. Two of the items showed significant differences between the gender groups and are reported here. Female students scored significantly lower on the item "I need to be a little better than most in the course to be happy with my performance" ($Mdn = 2$) than male students ($Mdn = 5$), $U = 130,5$, $p < .001$, $r = .57$ (Mann-Whitney test). Women scored higher ($Mdn = 5.5$) than men ($Mdn = 5.5$) on the item "If I work with someone who is better than me, I learn a lot" ($Mdn = 4$), $U = 517$, $p = .037$, $r = .28$ (Mann-Whitney test). All of the four differences between female and male are also displayed in box plots. See Figure 1.

TABLE IV
CORRELATION BETWEEN LEARNING STRATEGIES AND OTHER VARIABLES

	Individual thinker	***Social reader***	***Interactive problem-solver***
NFC			$r = .31$*
IMI (interest-enjoyment)	$r_s = .56$**		$r = .27$*
IMI (perceived competence)	$r_s = .49$**		$r = .27$*
IMI (tension-pressure)		$r = .28$*	
IMI (effort-importance)	$r_s = .43$**	$r_s = -.31$*	

r: Pearson's r - Pearson product-moment correlation coefficient.
r_s: Spearman's rho - Spearman's rangcorrelation coefficient.
**. Correlation is significant at the 0.01 level (2-tailed).
*. Correlation is significant at the 0.05 level (2-tailed).
$N = 61$ for NFC, $N = 58$ for IMI, all sub-scales

B. Correlations between learning strategies and other factors

The factor analysis revealed three distinct factors, as described above (see table I and II). The *Individual thinker* strategy is characterised by learning when testing an idea by writing code individually and when stuck, trying to solve the problem by oneself first. The strategy can be said to be more "thinking first, and then writing code". Contrasting, the *Interactive problem-solver* strategy is interactive, and thinking and writing code is intertwined, using other resources such as Google when needed. What is salient in the *Social reader* strategy is that working together with others, asking for help, discussing or looking at someone writing code, is the main way to learn. The *Social reader* strategy also includes reading lecture notes. Individual students use all these strategies to some extent, but to varying degrees. As a final step, we conducted correlation analyses where these learning strategy factors described above were analysed in relation to motivation (IMI) and need for cognition (NFC). The results from the correlation analyses are summarized in table IV. The *Interactive problem-solver* strategy correlated significantly with NFC, and the IMI subscales interest-enjoyment and perceived compe-

tence. The *Social reader* strategy correlated significantly with IMI subscale tension-pressure, and also with gender (described above). Notably is the negative correlation with the IMI subscale effort-importance. The *Individual thinker* strategy highly correlated with the IMI subscales effort-importance, perceived competence and interest-enjoyment. Thus, intrinsic motivation and an *Individual thinker* strategy are highly correlated taken as a whole. In addition to the NFC and IMI tests, we also tested emotional items about programming (constructed by us as described above) and found that the three strategies correlated with different items. The *Individual thinker* strategy correlated with feelings of pride and joy, and with plans to continue with programming because it was delightful (see table V). The *Social reader* strategy correlated with feelings of frustration and experiencing learning programming as an emotional roller-coaster, and negatively correlated with plans of continuing with programming after the course (see table VI). The *Interactive problem-solver* strategy did not correlate with any emotional items, but with plans to continue with programming because of perceived competence (see table VII).

TABLE V
CORRELATION THE STRATEGY *Individual thinker* AND ITEMS (SPEARMAN'S RHO, SIG. (2-TAILED))

Items	***Corr***	***Sig.***
It is educational to explain to others	$r_s(53) = .57$	$p < .001$
I am proud of my solutions	$r_s(53) = .43$	$p = .001$
I am most satisfied with a solution I have come up with completely by myself	$r_s(54) = .32$	$p = .018$
I intend to continue programming because it gives me pleasure	$r_s(52) = .34$	$p = .013$
I see programming as a language where I communicate with the computer	$r_s(53) = .28$	$p = .042$

TABLE VI
CORRELATION THE STRATEGY *Social reader* AND ITEMS (SPEARMAN'S RHO, SIG. (2-TAILED))

Items	***Corr***	***Sig.***
I need to be a little better than most in the course to be happy with my performance	$r_s(52) = -.43$	$p = .001$
If I work with someone who is better than me, I learn a lot	$r_s(53) = .35$	$p = .009$
I experience learning programming as an emotional roller coaster	$r_s(54) = .34$	$p = .011$
I intend to continue programming because it gives me pleasure	$r_s(52) = -.33$	$p = .014$
I intend to continue programming because I'm good at it	$r_s(52) = -.31$	$p = .023$
I get frustrated when my solutions do not work	$r_s(54) = .31$	$p = .021$
I am most satisfied with a solution I have worked out together with others	$r_s(52) = .29$	$p = .03$

VII. RELIABILITY AND VALIDITY

In this section, we will discuss both the validity and reliability of our study. There are several limitations to our study. First, the sample is relatively small and collected at one university only. Second, we collected the survey in connection to the mid-term exam, which was not an ideal point in time. Also, the survey included relatively many items, which may explain some internally missing data. Our relatively low response rate of 43% is due to not getting access to all the students, not biased of student choosing not to hand the survey in, which gives reliability to our results despite the response rate. However, the internal consistency of the established inventories used (NFC and IMI) were satisfactory. In table VIII the items of the inventories we used in our study were tested with a Cronbach's alpha value [39] of over .8 for all of the IMI subscales which is considered good, and .7 for NFC which is considered acceptable.

TABLE VII
CORRELATION THE STRATEGY *Interactive problem-solver* AND ITEMS (SPEARMAN'S RHO, SIG. (2-TAILED))

Items	***Corr***	***Sig.***
I see programming as a language where I communicate with myself	$r_s(53) = .48$	$p < .001$
I intend to continue programming because I'm good at it	$r_s(52) = .44$	$p = .001$
You do not have to be better at programming to be able to help someone	$r_s(53) = .36$	$p = .007$
Programming is very open and there are no clear right and wrong	$r_s(54) = .31$	$p = .027$
I need to be a little better than most in the course to be happy with my performance	$r_s(52) = .29$	$p = .032$
It is educational to explain to others	$r_s(53) = .29$	$p = .031$

TABLE VIII
INTERNAL CONSISTENCY (CRONBACH'S ALPHA)

Index	Nr. of items	Cronbach's alpha value
NFC	16	.75
IMI (interest-enjoyment)	7	.84
IMI (perceived competence)	6	.91
IMI (tension-pressure)	4	.82
IMI (effort-importance)	4	.80

Furthermore, the exploratory factor analysis revealed three distinct factors, which in turn correlated significantly to other variables. The factors, or learning strategies, relate to previous work [9] and the interviews with students taking this particular introductory course [7], [10]. Thus, the ecological validity of the identified factors is considered good.

VIII. DISCUSSION

In this study we report on three different strategies to learn programming: *Individual thinker*, *Social reader* and *Interactive problem-solver*. Two of these strategies, the *Individual thinker* and the *Interactive problem-solver*, highlight that learning programming involves both doing and thinking, but in slightly different ways. While the latter strategy seems to be a highly interactive and intertwined process of thinking and doing, the former suggests that the thinking and doing might be more separated so that students think about a solution first, and then implement that idea by writing code. These two strategies are in agreement with previous findings of learning programming by going back and forth between thinking and writing code,

e.g. between theory and practice [7], [13], [40]. The third strategy, the *Social reader*, highlights the social aspects of working and learning together in addition to the thinking and doing aspects.

The correlation analyses revealed interesting differences between the strategies regarding motivation. The *Individual thinker* and the *Interactive problem-solver* strategies correlated positively to being interested and enjoying programming, and feeling competent. The *Social reader* strategy, on the other hand, correlated positively with experiencing tensions and pressure. This indicates that students who used *Individual thinker* or *Interactive problem-solver* strategy was more internally motivated, and students who preferred the *Social reader* strategy were driven mainly by extrinsic motivation. Consequently, this may influence their learning, since intrinsic motivation have been found to positively correlate to higher achievement [26]. We also found that the *Interactive problem-solver* strategy correlated positively with a high score on the NFC inventory, suggesting that students who prefer an *Interactive problem-solver* strategy also enjoy thinking more compared to students who use the other two strategies. Although the tendency to enjoy and engage in thinking is considered to be relatively stable, young people have been found to increase their score on NFC over time [41]. One explanation to our finding might be that, since programming is cognitively demanding, learning programming and develop an interest and competence in programming also develops your interest and enjoyment in thinking and problem-solving.

Furthermore, we found differences in how the strategies related to emotional aspects. The *Individual thinker* strategy related to positive emotions, such as being proud and satisfied with own solutions, and wanting to continue with programming because it is pleasurable. Contrasting, the *Social reader* strategy related to negative emotions, such as frustration and the experience of learning programming as an emotional roller coaster. Solutions that were the result of collaborative work were more satisfying for students who preferred the *Social reader* strategy. Interestingly, the *Interactive problem-solver* strategy did not correlate to emotional aspects.

Our results also point to a strong gender difference, where female students seemed to prefer the *Social reader* strategy and reported to a lesser extent that they planned to continue with programming. Social aspects of learning programming seems to be important, especially to women. One explanation may be that due to lower computing self-efficacy and belief about their ability [4], women may find working together at the computer comforting, helpful and more enjoyable [10], [16]. Another explanation may be that female students value a collaborative learning environment, where helping others and contributing to the community is the norm, higher than men do [20]. It is discouraging that our results point in the same direction as other studies [4], [15], [20], indicating that women to a lesser extent plan to continue with programming after the introductory course.

The majority of the students in our study were novices, lacking previous experiences of programming. However, our findings suggest that the students with previous experiences of programming had a major advantage. Experienced students felt more competent, less pressured, more interested and enjoyed programming more compared to students without prior experience. Students with prior experience seem to use the *Interactive problem-solver* strategy more than inexperienced students. Interestingly, this strategy is less associated with emotions, suggesting that novices may experience more emotions when learning programming. Furthermore, it was also the experienced students who, to a greater extent, reported that they wanted to continue with programming. These findings raise questions on how well we facilitate learning for novices and female students in the introductory programming courses.

Our results indicate that experienced students use the *Interactive problem-solver* strategy, suggesting that this highly interactive way of communicating with the computer, practical thinking while programming, is something that develops with experience. In line with this, we suggest that novice students have developed *Individual thinker* and *Social reader* strategies before entering university. These strategies may be more general, applicable to learning in many subjects. Learning programming, on the other hand, may require a more interactive way of thinking and doing, and it takes time to develop this strategy. This explains why it was the experienced students who used the *Interactive problem-solver* strategy. Thus, we speculate that students who initially prefer the *Individual thinker* or the *Social reader* strategy can develop the *Interactive problem-solver* strategy when gaining more knowledge and programming experience. However, it is unclear how students who prefer and enjoy working together develop a strategy focused on individual problem-solving.

IX. Conclusion and future work

In this study, we explored differences in students' strategies when learning programming. We identified three distinct strategies: *Individual thinker*, *Social reader* and *Interactive problem-solver*. Gender and prior programming experience were related to these strategies, suggesting that novice students to a higher extent used *Individual thinker* and *Social reader* strategies, where female students preferred the latter. Allowing students to choose how to work, and who to work with, together with fostering a more collaborative, inclusive learning environment may be one way to encourage more women to feel that they belong, and choose to continue with programming. Our results suggest that learning programming involves developing practical thinking, a strategy where students interact with the computer, thinking and trying ideas while writing code, a strategy we call *Interactive problem-solver*. However, future research is needed to demonstrate if, and how, students develop their strategy over time as they learn more and become more competent.

Acknowledgments

We wish to thank all the students who participated in this study. This project is supported by The Swedish Research Council, grant 2015-01920.

References

[1] A. Robins, "Novice programmers and introductory programming," *The Cambridge Handbook of Computing Education Research, Cambridge Handbooks in Psychology*, pp. 327–376, 2019.

[2] P. Kinnunen and L. Malmi, "Why students drop out cs1 course?" in *Proceedings of the Second International Workshop on Computing Education Research*, 2006, p. 97–108.

[3] A. Petersen, M. Craig, J. Campbell, and A. Tafliovich, "Revisiting why students drop cs1," in *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, 2016, p. 71–80.

[4] S. Beyer, "Why are women underrepresented in computer science? gender differences in stereotypes, self-efficacy, values, and interests and predictors of future cs course-taking and grades," *Computer Science Education*, vol. 24, no. 2-3, pp. 153–192, 2014.

[5] K. von Hausswolff, A. Eckerdal, and M. Thuné, "Learning to program hands-on: a controlled study," in *Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*, 2020, pp. 1–10.

[6] L. J. Höök and A. Eckerdal, "On the bimodality in an introductory programming course: An analysis of student performance factors," in *2015 International Conference on Learning and Teaching in Computing and Engineering*, 2015, pp. 79–86.

[7] K. von Hausswolff, "Practical thinking while learning to program – novices' experiences and hands-on encounters," Submitted.

[8] A. Eckerdal, "Relating theory and practice in laboratory work: A variation theoretical study," *Studies in Higher Education*, vol. 40, no. 5, pp. 867–880, 2015.

[9] R. McCartney, A. Eckerdal, J. E. Mostrom, K. Sanders, and C. Zander, "Successful students' strategies for getting unstuck," in *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 2007, p. 156–160.

[10] K. von Hausswolff and M. Weurlander, "Social dimensions in the lab session when novices learn to program," in *2020 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2020, pp. 1–9.

[11] K. von Hausswolff, "Practical thinking in programming education," in *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, 2017, pp. 203–204.

[12] V. Renumol, D. Janakiram, and S. Jayaprakash, "Identification of cognitive processes of effective and ineffective students during computer programming," *ACM Trans. Comput. Educ.*, vol. 10, no. 3, 2010.

[13] A. Eckerdal, R. McCartney, J. E. Moström, K. Sanders, L. Thomas, and C. Zander, "From limen to lumen: computing students in liminal spaces," in *Proceedings of the third international workshop on Computing education research*. ACM, 2007, pp. 123–132.

[14] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," in *Acm Sigcse Bulletin*, vol. 37, no. 3. ACM, 2005, pp. 14–18.

[15] A. Luxton-Reilly, I. Albluwi, B. A. Becker, M. Giannakos, A. N. Kumar, L. Ott, J. Paterson, M. J. Scott, J. Sheard, and C. Szabo, "Introductory programming: a systematic literature review," in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 2018, pp. 55–106.

[16] B. Hanks, S. Fitzgerald, R. McCauley, L. Murphy, and C. Zander, "Pair programming in education: A literature review," *Computer Science Education*, vol. 21, no. 2, pp. 135–173, 2011.

[17] F. J. Rodríguez, K. M. Price, and K. E. Boyer, "Exploring the pair programming process: Characteristics of effective collaboration," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE 17. New York, NY, USA: Association for Computing Machinery, 2017, p. 507512. [Online]. Available: https://doi.org/10.1145/3017680.3017748

[18] C. M. Lewis and N. Shah, "How equity and inequity can emerge in pair programming," in *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ser. ICER 15. New York, NY, USA: Association for Computing Machinery, 2015, p. 4150. [Online]. Available: https://doi.org/10.1145/2787622.2787716

[19] Lan Cao and Peng Xu, "Activity patterns of pair programming," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, 2005, pp. 88a–88a.

[20] L. J. Sax, J. M. Blaney, K. J. Lehman, S. L. Rodriguez, K. L. George, and C. Zavala, "Sense of belonging in computing: the role of introductory courses for women and underrepresented minority students," *Social Sciences*, vol. 7, no. 8, p. 122, 2018.

[21] J. D. Vermunt, "Relations between student learning patterns and personal and contextual factors and academic performance," *Higher Education*, vol. 49, no. 3, pp. 205–234, 2005.

[22] J. D. Vermunt and V. Donche, "A learning patterns perspective on student learning in higher education: State of the art and moving forward," *Educ Psychol Rev*, vol. 29, no. 2, pp. 269–299, 2017.

[23] N. Entwistle, *Teaching for understanding at university: Deep approaches and distinctive ways of thinking*. Basingstoke: Palgrave Macmillan, 2009.

[24] K. Trigwell, R. A. Ellis, and F. Han, "Relations between students' approaches to learning, experienced emotions and outcomes of learning," *Studies in Higher Education*, vol. 37, no. 7, pp. 811–824, 2012.

[25] A. Yacob and M. Saman, "Assessing level of motivation of learning programming among engineering students," in *Proceedings of the International Conference on Informatics and Applications (ICIA)*, 2012, pp. 425–432.

[26] M. Pedaste, O. Must, G. Silm, K. Täht, K. Kori, Leijen, and M.-L. Mägi, "How do cognitive ability and study motivation predict the academic performance of IT students?" in *Proceedings of ICERI2015 Conference, Sevilla, Spain*, 2015, pp. 7167–7176.

[27] K. Kori, M. Pedaste, Leijen, and E. Tõnisson, "The role of programming experience in ict students' learning motivation and academic achievement," *Int. J. Inf. Educ. Technol.*, vol. 6, no. 5, pp. 331–337, 2016.

[28] G. Biesta and N. C. Burbules, *Pragmatism and Educational Research*, 2003.

[29] J. Dewey, *Experience and Nature*, 1925/1995.

[30] K. von Hausswolff and A. Eckerdal, "Measuring programming knowledge in a research context," in *2018 IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018, pp. 1–9.

[31] L. Östman, K. Van Poeck, and J. Öhman, "A transactional theory on sustainability learning," in *Sustainable Development Teaching: Ethical and Political Challenges*. Routledge, 2019, pp. 127–139.

[32] P.-O. Wickman, *Aesthetic experience in science education: Learning and meaning-making as situated talk and action*. Routledge, 2006.

[33] J. Dewey, "Human nature and conduct: An introduction to social psychology," *Journal of Philosophy*, vol. 19, no. 17, pp. 469–475, 1922.

[34] J. T. Cacioppo and R. E. Petty, "The need for cognition," *J. Pers. Soc. Psychol*, vol. 42, pp. 116–131, 1982.

[35] J. Grass, A. Strobel, and A. Strobel, "Cognitive investments in academic success: The role of need for cognition at university," *Frontiers in Psychology*, vol. 8, p. 790, 2017.

[36] R. M. Ryan and E. L. Deci, "Intrinsic and extrinsic motivations: Classic definitions and new directions," *Contemporary Educational Psychology*, vol. 25, no. 1, pp. 54–67, 2000.

[37] E. McAuley, T. Duncan, and V. V. Tammen, "Psychometric properties of the intrinsic motivation inventory in a competitive sport setting: A confirmatory factor analysis," *Research quarterly for exercise and sport*, vol. 60, no. 1, pp. 48–58, 1989.

[38] J. T. Cacioppo, R. E. Petty, and C. Feng Kao, "The efficient assessment of need for cognition," *Journal of personality assessment*, vol. 48, no. 3, pp. 306–307, 1984.

[39] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297–334, Sep 1951. [Online]. Available: https://doi.org/10.1007/BF02310555

[40] A. Berglund and A. Eckerdal, "Learning practice and theory in programming education: Students' lived experience," in *2015 International Conference on Learning and Teaching in Computing and Engineering*, 2015, pp. 180–186.

[41] J. Bruinsma and R. Crutzen, "A longitudinal study on the stability of the need for cognition," *Personality and Individual Differences*, vol. 127, pp. 151–161, 2018.