Undergraduate students' learning approaches and learning to program.

Melanie Coles
Bournemouth University
mcoles@bournemouth.ac.uk

Keith Phalp
Bournemouth University
kphalp@bournemouth.ac.uk

Abstract
This study uses the Revised Two Factor Study Process Questionnaire (R-SPQ-2F) to explore undergraduate students' approaches learning to program. The expectation being that students using deep learning approaches will gain higher programming grades than students who use surface approaches. There is strong evidence to support the hypothesis that deep approaches are related to higher grade outcomes, and surface approaches to lower. There is also strong evidence to support the hypothesis that students who '*hate'* programming do less well than those that do not. There is however, no evidence that previous programming experience has an impact upon the student programming grade.

1. Introduction
Discussions about the difficulties involved in learning to program and the best way to teach programming have been part of the research field for decades, with educators reporting difficulties and failure, and dropout rates being high for programming courses (Bennedsen & Caspersen, 2007; Dijkstra, 1982; Hare, 2013; Jenkins, 2002; Mavaddat, 1976; Robins, Rountree, & Rountree, 2003; Simon et al., 2009; Watson & Li, 2014). Recognition that computer programming appears difficult for a high percentage of students, that many students settle for a pass grade, that students grasp programming principles (if ever) at widely varying times and that a very small percentage of students perform extremely well and demonstrate a keen interest in computing, have been reported since programming teaching began in the 60s (Mavaddat, 1976).

If programming is difficult to learn then a corollary of this is a higher failure rate for programming than for other undergraduate subjects. It is an often cited outcome that learning to program is notoriously difficult (Bornat, Dehnadi et al. 2008; Jenkins 2002, Robins et al 2003) however only a few papers fully explore the suggested higher failure rate and attempt to develop evidence to support this supposition (Bennedsen & Caspersen, 2007; Watson & Li, 2014). The findings of both papers suggest the majority of pass rates are in the range 50-80%, with an average of 67.7%.

Learning to programming involves a range of related, but also contentious elements, all of which need to align should a student hope to do well. Jenkins identified a number of factors that relate specifically to the domain of programming and what makes it difficult to learn rather than to the more commonly explored student aptitude for learning to program. Such factors involve the multiple skills and processes required, the language used to teach the students, the educational novelty of students learning to program, the student interest, the image and the pace of teaching (Jenkins, 2002). Many factors intertwine and have an impact upon the student individually: motivation, previous experience, time spent programming, aptitude for programming and student attendance. Environmental factors such as the teaching style, the programming language used, the assessment mechanism and a range of pedagogical interventions can also influence student performance. This paper focuses on and explores students' motivation and learning approaches when studying programming, and also includes students' emotional response to programming and their previous background.

2. Background

Understanding students' approaches to learning and exploring how such approaches relate to the module outcomes for students is clearly valuable information. If we can understand and impact upon students' motivation, can we influence students' success rates? Many interventions used in the teaching of programming may work because they alter the students' motivation and even that an intervention itself is taking place, may alter the students' motivation.

2.1 Motivation and Learning Approaches

Students' motivation towards their studies seems an obvious factor that could impact upon their outcomes on any module regardless of subject. If a student is motivated to succeed there is more likelihood of them achieving that success. However could motivation play a greater role in students learning to program than it does in other courses? Programming needs persistence and practice, students must be motivated to spend time practicing, even if there is no explicit assignment (Jenkins, 2001). The combination of students' motives to learn and the strategy they use determines their learning approach (Everaert, Opdecam, & Maussen, 2017).

Motivation is an abstract concept that is difficult to measure in any meaningful way (Jenkins, 2001), behaviour can be observed or questions can be asked but the true motivation behind behaviour is never certain. Jenkins results showed that the main motivators for students were firstly aspiration, but closely followed by the desire to learn, both classed as extrinsically motivated, rather than the intrinsic motivation of interest in the subject itself. There was little evidence from any of Jenkins' questions that students were interested in programming, with almost 50% of students only doing programming because it was compulsory – something that he cites (*and is probably backed up by most programming instructors*) as a depressing observation (Jenkins 2001).

Students who are more intrinsically motivated are found to perform better, with higher levels of intrinsic motivation leading to higher programming results (Bergin and Reilly 2005). Students with intrinsic motivation usually undertook to learn programming in their own time, had prior programming experience and displayed higher capabilities. Such students engaged in programming meaningfully, showed persistence in *playing with code* and would apply what they had learnt to real world problems, compared to others who approached their work in a more trial-and-error or impulsive fashion (Carbone, Hurst, Mitchell, & Gunstone, 2009). Bergin and Reilly suggest that extrinsic motivation does not appear to impact upon results, so suggesting that the use of grades, rewards or student comparisons are not useful for motivating students and that educator efforts should focus on improving students intrinsic motivation (Bergin and Reilly 2005).

Carbone et al found that students could experience a change in motivation, they could start off intrinsically motivated but then experience a change so becoming extrinsically motivated and vice-versa. This change in motivation could be triggered by a range of factors including: no reward for extra effort, encountering difficulties they could not resolve, perceived waste of time on tasks, and lack of technical skills. The technical skills were further catalogued into: an inability to identify problems, ineffective tinkering, inability to break programming problem down, lack of problem solving skills, and limited debugging skills. Carbone et al also identified some personal skills that impacted upon students' motivation: poor time management, independence (over reliance on others) and attitude toward programming errors (Carbone et al., 2009). It is interesting how changeable and sensitive motivation appears to be to external factors, such that reward (*in the form of a grade*) could alter a student's motivation (*in both directions*), how undertaking additional effort and perceiving no reward (*again from the marker*) could impact negatively on a student's motivation.

2.2 Emotion

The student's emotional response to learning to program has not received much research attention (Chetty & Van der Westhuizen, 2013) possibly due to the scientific, engineering domain and the stereotypical lack of emotion in these subject area. The stereotype associated with the logical approach, for example Mr Spock from Star Trek, seems to exist in isolation from emotion, yet it is evident from interacting with students learning to program that they experience a range of strong emotions. They "hate programming", they "love programming", they find it "frustrating", "challenging", "rewarding" all of which indicate a strong emotional response. It would seem an obvious corollary that such emotion would have an impact upon the student's motivation and so their programming performance. More successful students appear to have a more positive view of programming (Simon et al., 2009), and whilst this does seem evident the further question maybe - is it the higher grades that promote the liking or the liking that promotes the higher grades?

Simon et al in a survey of 697 students enrolled in seven courses at five institutions found that nearly half (48%) of the 2553 comments received where classified as positive. The two most positive categories listed by students were using the words *fun/cool* or *interesting/rewarding*. Nearly a third of the comments made were negative (32%), with the most often response being *hard/difficult* and *frustrating/stressful* (Simon et al., 2009). Does this third that make negative comments go in some way to explain the high failure rate of programming undergraduates; is there a link between this negative emotional response and a lower grade?

Many of my students say things like programming is "*tough but rewarding*", "*very difficult*", "*too complicated*" and the one I have heard the most often "*I hate programming*". Emotions can profoundly affect students' thoughts, motivation and action, positive emotions such as enjoyment of learning may generally enhance academic motivation. Although negative emotions are not always detrimental, for example task-related anger may trigger motivation to overcome obstacles (Pekrun, Goetz, & Titz, 2002).

2.3 Previous background

One of the most important variables affecting general university performance is past academic results (Alam, Billah, & Alam, 2014). Byrne and Lyons found some significance both in student's mathematics and science results from their Irish Leaving Certificate and their programming examination score, although no such significance was found with English or Foreign Language results (Byrne & Lyons, 2001). The higher grades in both maths and science correlated with students programming scores. Other studies have also found that a maths background correlates with students programming performance (Cantwell-Wilson & Shrock, 2001). So is it that students who have an aptitude for science and maths also have an aptitude for programming or is it that the students who undertook the maths and science (an option) were better prepared to succeed at programming?

What about students' previous exposure to programming, a logical conclusion is that students who could already program would do better than those who had not studied it before. Research does seem to support his suggestion, that experience with programming does benefit students (Hagan & Markham, 2000), but the specific language experienced may be the important factor (de Raadt, Hamilton, Lister, & Tutty, 2005).

3. Hypotheses

Following on from the initial literature review four main hypotheses were developed:

H1: Students with a deep approach to learning will gain higher grades in programming than students with a surface approach

H2: Students with a surface approach to learning will gain lower grades than students with a deep approach

H3: Students who can already program or who have studied a programming before starting university will gain higher grades than students who have not.

H4: Students who have negative emotions towards programming will gain lower grades than students who do not.

4. Methodology

4.1 Instrument Used

The Revised Two Factor Study Process Questionnaire (R-SPQ-2F) was used, this questionnaire is suitable for use to evaluate how students learn or how they approach learning. The revised version of the questionnaire has two main scales Deep Approach (DA) and Surface Approach (SA) with four sub-scales: Deep Motive (DM), Deep Strategy (DS), Surface Motive (SM) and Surface Strategy (SS), shown in the Table 1 (Biggs, Kember, & Leung, 2001; de Raadt et al., 2005). Students adopting a surface approach build their view from facts and details of activities with the aim of reproducing material rather than making theoretical connections, while those adopting a deep learning approach seek to understand the material they are studying.

	Surface	**Deep**
Motive	fear of failure, emphasis is external, from demands of the assessment	intrinsic interest, emphasis is internal
Strategy	narrow target, rote learn memorises information	maximise meaning relates knowledge

Table 1: From Biggs (2001) and de Raadt (2005)

R-SPQ-2F was used, but rather than the generic form it was modified to apply specifically to learning to programming, thus

1. I find that at times studying gives me a feeling of deep personal satisfaction

becomes

1. I find that at times studying programming gives me a feeling of deep personal satisfaction.

and

7. I do not find my course very interesting so I keep my work to the minimum.

becomes

7. I do not find my programming unit very interesting so I keep my work to the minimum.

This was to focus the questionnaire specifically on programming rather than on general strategies. As the strategies used would be expected to differ for different disciplines studied. Additional questions were also added to the questionnaire to explore students' previous experience with programming

I can already program

I have completed a programming course (at school or college)

A further question was added to explore students' general emotional response to programming:

I hate programming.

This question was used as it is the most used by the students themselves.

All questions had a five point Likert Scale response, using alpha characters:

A — this item is *never* or only *rarely* true of me
B — this item is *sometimes* true of me
C — this item is true of me about *half the time*
D — this item is *frequently* true of me
E — this item is *always* or *almost always* true of me

4.2 Process

The questionnaires were issued to all students present in lectures and seminars on third week of term, so students had only had three weeks of teaching. There were 293 students on the course, of these

- 36 students did not complete both the coursework and the exam, for a variety of reasons and these were removed from the study
- 121 students completed the questionnaire and both the coursework and the exam
- 136 did not complete the questionnaire, or did not complete it fully (*no signature or not all questions answered*). Some students were present but elected not to complete it, others were not present.

The students all undertook the same module (*unit in our terminology*); Principles of Programming (PoP), which is an introductory programming unit, taught in the first semester of the students' first year, no previous programming knowledge was assumed. The students had a two hour lecture and a two hour lab session each week, for 12 weeks. These lectures covered a foundational programming topic, starting with variables and data manipulation, then selection, loops, file reading and writing and finishing with sorting and searching. For the coursework students had to upload multiple tasks every other week (*four different sets of tasks*), and the end of the 12 week block there was an exam. The four pieces of coursework together give 50%, with the earlier ones being weighted less (5%, 5%, 20% and 20%) of the overall unit total and the exam gives the other 50%.

All questionnaires were then put away until after the module had finished.

4.3 Threats to validity

Some students either elected to not complete the questionnaires or were not present when the questionnaires were issued and such self-selection may have an impact upon findings. Was there a difference in achievement between the students who completed the questionnaire and those that did not? The analysis can be seen in table 2 below.

	Coursework	**Exam**	**Unit Total**
Completed	71.0	71.9	71.4
Did not complete	60.6	61.6	61.1

Table 2: Unit Averages

There was a difference for both coursework and exam scores individually and also obviously for the unit total. This may indicate the difference in attendance vs non-attendance in the unit outcomes for the students, those not attending are already engaging in behaviour that may impact negatively on their grades. Students who were present and elected to not complete the questionnaire may be those less interested in the academic discipline and helping with research, or possibly more concerned about the relationship of completing the questionnaire to their programming marks.

As students are self-reporting what they say their approach to learning is and what it really is may differ. Also the fact that they were completing the research study for one of their unit tutors may impact upon their responses to questions, they may have responded as they thought was '*best'*, even though students were assured the questionnaires would not be looked at until after they had finished the unit.

5. Results

The first exploration of the results was to correlate the response to the overall unit average as can be seen in Table 3 below.

Approach	Correlation	Significance
deep approach	0.3374	**.000154**
surface approach	-0.3584	**.000055**
I could already program before starting university	0.1432	*.11713*
I had completed a programming course before starting university	0.1077	*.239664*
I hate programming	-0.3269	**.000263**

Table 3: Correlation of questionnaire answers to unit total

So the deep approach is positively and significantly correlated with the student's unit total and surface approach is negatively correlated, both of which support the H1 and H2 hypotheses. However what is interesting is that neither the students' (reported) ability to be able to program or their having previously studied programming had a significant correlation with the unit total. So not supporting the H3 hypothesis. Students' emotional response, the "*I hate programming"* question, is also negatively correlated with the unit total, so supporting hypothesis H4.

5.1 Further Analysis

A more detailed analysis of the data was explored, examining the relationship between each of the sub-scales and the unit assessment element, i.e. either coursework or exam; this can be seen in Table 4 below.

Approach	Against	Correlation	Significance
Difference between deep - surface	unit total	0.3968	**.00001**
deep motive	coursework	0.3592	**.000052**
deep motive	exam	0.3139	**.000455**
deep motive	unit total	0.3760	**.000021**
deep strategy	coursework	0.1569	*.085679*
deep strategy	exam	0.2245	***.013303***
deep strategy	unit total	0.2165	***.017073***
surface motive	coursework	-0.2636	**.003563**
surface motive	exam	-0.3307	**.000219**
surface motive	unit total	-0.3359	**.000173**
surface strategy	coursework	-0.2142	***.018426***
surface strategy	exam	-0.2886	**.001359**
surface strategy	unit total	-0.2849	**.001595**

Table 4 : Correlation of sub-elements to unit assessment

As the questions for both the deep and the surfaces approaches could all be scored at either A or E, the difference between the scores was calculated (DA minus SA) to see if the difference would also

correlate to student outcomes. As can be seen in Table 4 above this proved significant, so whilst some students may just have been entering As and Bs almost at random there was evidence that students with a high deep approach score and low surface approach score would have improved outcomes in programme.

Looking at the different groups of questions that make up the deep or surface approaches there are more nuanced results. It is interesting to note that there is strong evidence for a relationship between deep motive and coursework, exam and unit total outcomes, which are all significant. However the correlation between deep strategy and coursework is not statistically significant, and there is weaker significance for deep strategy and both exam and unit total. Also of note is that there is only weak evidence for surface strategy to coursework.

6. Discussion

6.1 Deep Approaches

So clearly having an intrinsic interest in programming improves outcomes for students on an undergraduate programming course. Although it could be that those attending and so filling in the questionnaire were more likely to be those interested in programming. However the deep strategies employed appear to have less of an impact upon the outcomes for students. Deep strategy: the five questions that make up this group are:

- *I find that I have to do enough work on a programming topic so that I can form my own conclusions before I am satisfied.*
- *I find most new programming topics interesting and often spend extra time trying to obtain more information about them.*
- *I test myself on important programming topics until I understand them completely.*
- *I spend a lot of my free time finding out more about interesting programming topics which have been discussed in different classes.*
- *I make a point of looking at most of the suggested readings that go with the programming lectures.*

Is it just that such strategies do not apply entirely to programming as a subject? Whilst the last question, following up on reading, is possibly not high on a programmers list of strategies as the staff, the internet and other students are possibly more likely to be used as a source of support. The other four questions are all stereotypical behaviours associated with the archetypal programmer: writing extra code and *playing with code* until you understand it, writing code for fun in spare time. It is interesting that these results suggest that such behaviour is not important to university outcomes for coursework and exams in programming.

6.2 Surface Strategy

Surface strategy when applied to coursework does not appear to be negatively correlated. The five questions that make up this group are:

- *I only study seriously what's given out in class or in the course outlines.*
- *I learn some things by rote, going over and over them until I know them by heart even if I do not understand them.*
- *I generally restrict my programming study to what is specifically set as I think it is unnecessary to do anything extra.*
- *I believe that lecturers shouldn't expect students to spend significant amounts of time studying programming material everyone knows won't be examined.*
- *I find the best way to pass examinations is to try to remember answers to likely questions*.

Such approaches clearly do not have an impact on the quality of the coursework. The coursework submitted by students does focus on a particular topic being covered that week – so for example of writing loops, potentially the assessment used does not suffer when surface approaches are used? Strategy as applied to such coursework would not particularly suffer from a surface approach as once done the student moves onto the next task, that is they become task focused.

6.3 Previous Experience

Previous experience does not impact on unit outcomes, this suggests that approach to study is more important than previous experiences. Previous experience may also have an impact upon both learning approaches and upon emotion before starting their undergraduate course. Many students have anecdotally reported poor experiences of programming at school or college, therefore previous experience could also negatively impact upon results.

6.4 I hate programming

Possibly it is of no surprise that emotional response is negatively correlated with performance. The questionnaire was distributed early in the unit, therefore the response cannot be related to students' coursework grades as they had not yet been returned or potentially to the perceived difficulty of the unit as students do tend to be comfortable with the concept for the first few weeks of programming. However students who had met programming at school previously and who had struggled with it may already have a negative emotional response to programming that does impact upon their unit grades.

6.5 Relationship to Other Subjects

Whilst the questionnaire was focused on student approaches to programming, and there is evidence that the approaches adopted by students do impact upon their programming grades, the obvious further question is … do these approaches apply to all subjects?

Approach	Against	Correlation	Significance
deep approach	Application of Programming	0.3182	**0.00045**
surface approach	Application of Programming	-0.3230	**0.00034**
deep approach	Networks and Cyber Security	0.1611	*0.08157*
surface approach	Networks and Cyber Security	-0.3481	**0.00011**
deep approach	Systems Analysis and Design	0.1107	*0.23315*
surface approach	Systems Analysis and Design	-0.3053	**0.00078**
deep approach	Computer Fundamentals	0.1826	***0.04781***
surface approach	Computer Fundamentals	-0.2875	**0.00163**
deep approach	Data and Databases	0.1830	***0.04734***
surface approach	Data and Databases	-0.2392	**0.00915**

Table 5: Correlation of questionnaire answers to other unit totals

There is still strong evidence for deep approaches correlating positively to the second semester programming unit, there is only weak or no evidence for deep approaches correlating to other subjects. Not necessarily to be unexpected as the questionnaire was specifically focused on programming. However what is interesting is the strong evidence that surface approaches negatively relate to all unit outcomes. This suggests that students who use surface approaches for programming use such strategies for all units, and that this has an impact upon their success at university.

7. Conclusion

So whilst there is strong evidence for H1, H2 and H4 from the analysis of the questionnaires, there is no evidence to support H3. This does lead to further questions:

- Is there a relationship between previous experiences of programming and emotional response?
- How can students' approaches to learning be impacted by the tutors?
- Can different assessment strategies impact upon students' approaches?
- Could explicit discussion of students' approaches help them adopt more useful strategies?

Motivation and the learning approaches used by students do appear to impact upon their success rates on an introductory programming module. Deep approaches have a positive impact specifically on programming grades, there is less evidence for other subjects. Surface approaches have a negative impact upon grades for programming and also extend across other subjects, evidencing that such strategies are to the detriment of student performance.

8. References

Alam, M. M., Billah, M. A., & Alam, M. S. (2014). *Factors Affecting Academic Performance of Undergraduate Students at International Islamic University Chittagong* (IIUC), Bangladesh. Journal of Education and Practice, Vol.5(No.39).

Bennedsen, J., & Caspersen, M. E. (2007). *Failure Rates in Introductory Programming*. The SIGCSE Bulletin, Volume 39(Number 2).

Bergin, S. and R. Reilly (2005). *The influence of motivation and comfort-level on learning to program*. Psychology of Programming Interest Group (PPIG 17). Sussex University.

Biggs, J., Kember, D., & Leung, D.Y.P. (2001). *The Revised Two Factor Study Process Questionnaire: R-SPQ-2F*. British Journal of Educational Psychology, 71, 133-149.

Byrne, P., & Lyons, G. (2001). *The Effect of Student Attributes on Success in Programming*. Paper presented at the ITiCSE, Canterbury, UK.

Cantwell-Wilson , B., & Shrock, S. (2001). *Contributing to Success in an Introductory Computer Science Course: A Study of Twelve Factors Paper* presented at the ACM SIGCSE 2001, Charlotte, NC, USA.

Carbone, A., Hurst, J., Mitchell, I., & Gunstone, D. (2009). *An Exploration of Internal Factors Influencing Student Learning of Programming.* Paper presented at the Australasian Computing Education Conference (ACE 2009), Wellington, New Zealand.

Chetty, J., & Van der Westhuizen, D. (2013). *" I hate programming " and Other Oscillating Emotions Experienced by Novice Students Learning Computer Programming*. Paper presented at the EdMedia: World Conference on Educational Media and Technology, Victoria, Canada.

de Raadt, M., Hamilton, M., Lister, R., & Tutty, J. (2005). *Approaches to learning in computer programming students and their effect on success*. Paper presented at the Higher education in a changing world Research and development in higher education, Sydney, Australia.

Dijkstra, E. W. (1982). *How do we tell truths that might hurt*? ACM SIGPLAN Notices, 17(5), 13-15. doi:10.1145/947923.947924

Everaert, P., Opdecam, E., & Maussen, S. (2017). *The relationship between motivation, learning approaches, academic performance and time spent*. Accounting Education. doi:10.1080/09639284.2016.1274911

Hagan, D., & Markham, S. (2000). *Does It Help to Have Some Programming Experience Before Beginning a Computing Degree Program*? Paper presented at the ITiCSE 2000, Helsinki, Finland

Hare, B. K. (2013). *Classroom Interventions To Reduce Failure & Withdrawal In Cs1 – A Field Report* Journal of Computing Sciences in Colleges Volume 28(Issue 5).

Hawi, N. (2010). *Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course.* Computers & Education, Volume 54.

Jenkins, T. (2001). *The Motivation of Students Programming*. Paper presented at the ITICSE, Canterbury, UK.

Jenkins, T. (2002). *On the Difficulty of Learning to Program.* Paper presented at the LTSN-ICS Conference, Loughborough University.

Mavaddat, F. (1976). *An experiment in teaching programming languages*. ACM SIGCSE Bulletin, 8(2), 45-59. doi:10.1145/382220.382470

Pekrun, R., Goetz, T., & Titz, W. (2002). *Academic Emotions in Students' Self-Regulated Learning and Achievement: A Program of Qualitative and Quantitative Research*. Educational Psychologist.

Robins, A., Rountree, J., & Rountree, N. (2003). *Learning and Teaching Programming: A Review and Discussion*. Computer Science Education, Vol 13(No 2.), p 137-172.

Simon, B., Hanks, B., McCauley, R., Morrison, B., Murphy, L., & Zander, C. (2009). *For me, programming is ...* Paper presented at the ICER '09, Berkeley, California.

Watson, C., & Li, F.W.B. (2014, 06/21/2014). *Failure rates in introductory programming revisited*. Paper presented at the Proceedings of the 2014 conference on Innovation & technology in computer science education.