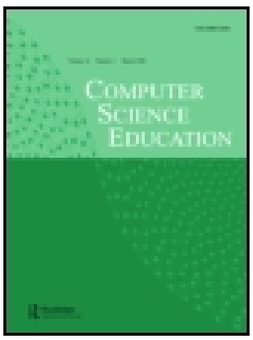

Computer Science Education

ISSN: 0899-3408 (Print) 1744-5175 (Online) Journal homepage: http://www.tandfonline.com/loi/ncse20

Predicting introductory programming performance: A multi-institutional multivariate study

Susan Bergin & Ronan Reilly

To cite this article: Susan Bergin & Ronan Reilly (2006) Predicting introductory programming performance: A multi-institutional multivariate study, Computer Science Education, 16:4, 303-323, DOI: 10.1080/08993400600997096

To link to this article: http://dx.doi.org/10.1080/08993400600997096

Computer Science Education
Vol. 16, No. 4, December 2006, pp. 303–323

Routledge
Taylor & Francis Group

Predicting Introductory Programming Performance: A multi-institutional multivariate study

Susan Bergin* and Ronan Reilly
Department of Computer Science, NUI Maynooth, Co. Kildare, ROP

A model for predicting student performance on introductory programming modules is presented. The model uses attributes identified in a study carried out at four third-level institutions in the Republic of Ireland. Four instruments were used to collect the data and over 25 attributes were examined. A data reduction technique was applied and a logistic regression model using 10-fold stratified cross validation was developed. The model used three attributes: Leaving Certificate Mathematics result (final mathematics examination at second level), number of hours playing computer games while taking the module and programming self-esteem. Prediction success was significant with 80% of students correctly classified. The model also works well on a per-institution level. A discussion on the implications of the model is provided and future work is outlined.

1. Introduction

Student retention on third-level Computer Science (CS) and Information Technology (IT) courses is a significant problem. Students find computer programming difficult and struggle to master the core concepts. Identifying struggling students can be difficult as introductory programming modules tend to have a very high student to lecturer ratio and thus lecturers do not know how well students are doing until after the first assessment. Given the typically high number of students, marking the assessments can take a considerable length of time. Even if the assessment is indicative of likely overall performance on the module, it may be too late for students to withdraw from the course or for instructors to intervene to prevent struggling students from failing. This is a cause of great concern for computer science educators and has led to a body of research in the area. Although many studies have interesting results it can be hard to know how to apply the results to other educational settings with different parameters, for example language being taught and assessment structure. Furthermore, the factors examined are often dependent upon the students'

*Corresponding author. Department of Computer Science, NUI Maynooth, Maynooth, Co. Kildare, ROI. E-mail: sbergin@cs.nuim.ie

ISSN 0899-3408 (print)/ISSN 1744-5175 (online)/06/040303-21 © 2006 Taylor & Francis
DOI: 10.1080/08993400600997096

experience on the module and with the material and therefore it is difficult to know how predictive the factors would be if measured at the commencement of the module.

A model that could predict likely programming performance in the first few weeks of a module would considerably help to alleviate this problem. To build such a model would require (1) the identification of early predictors of performance on an introductory programming module and (2) the appropriate implementation of a scientifically sound, predictive statistical model.

This paper describes such a model. A study on early identifiable factors that influence performance on an introductory programming module is presented and a model using these factors is developed. The paper is structured as follows. First, a literature review is provided, followed by a description of our study on early identifiable factors. Then the procedures used to prepare the data for analysis are outlined and subsequent analysis and results are presented. A discussion of the findings is presented and a brief outline of an epilogue study confirming the main findings is provided. The paper concludes with suggestions for future work.

2. Review of Literature

Predictors of performance on introductory programming modules studied to date can be broadly classified into the following groups: previous academic and computer experience with emphasis on exposure to mathematics and prior programming experience; cognitive factors; and psychological factors with emphasis on perceived comfort-level on the course. Studies within each of these categories are presented next and a brief summary of some other interesting factors that do not fall into the previous categories are also provided. Table 1 summarizes the predictors examined. Finally, another factor, self-regulated learning, that has previously been found to relate to performance in other academic domains is introduced and briefly discussed.

Previous academic experience and programming experience have often been cited as predictors of programming success. Numerous studies, including Leeper and Silver (1982), Honour-Werth (1986), Byrnes and Lyons (2001), Cantwell-Wilson and Shrock (2001), Evans and Simkin (1989), and Bergin and Reilly (2005a), have found that mathematical ability and exposure to mathematics courses are important predictors of performance on introductory computer science modules. Similarly, Byrnes and Lyons (2001) and Leeper and Silver (1982) found that performance in and experience of science subjects is also important. Studies by Hagan and Markham (2000), Holden and Weeden (2004), Evans and Simkin (1989), and Cantwell-Wilson and Shrock (2001) have ascertained that prior programming experience and non-programming computer experience are useful predictors of programming performance.

The role of cognitive factors in programming has also received research attention. Austin (1987), Barker and Unger (1983), Gibbs (2000), Hostetler (1983), Kurtz (1980), and Mayer et al. (1986) have investigated certain cognitive factors, including cognitive style and abstract reasoning ability and provide useful insights into the role of cognition in learning to program.

Table 1. Previous research on factors that influence programming performance

Researchers	Language	n	Reference criterion	Significant predictors
Austin (1987)	Pascal	76	Composite score based on program reading, program writing and lab performance	A model composed of high school composite achievement, quantitative and algorithmic reasoning abilities, vocabulary and general information abilities, self-assessed mathematics ability and measures of an introverted/analytic style and extroverted level was able to account for 64% of the variance in results
Barker and Unger (1983)	C++ and C	353	Final grade of students in 15 class sections learning two different programming languages	Intellectual development (ID) test successfully predicted advanced students. A regression model based on ID level accounted for 12% of the variance in final course grade
Bergin and Reilly (2005a)	Java	80	Overall module mark on an introductory programming course (30% continuous assessment and 70% final examination)	A regression module, based upon a student's perception of his or her understanding of the module, gender, Leaving Certificate (LC) mathematics score and comfort level was able to account for 79% of the variance in programming performance results
Bergin and Reilly (2005b)	Java	57	Early laboratory examination on an introductory programming module	Significant correlations found for: intrinsic motivation and self-efficacy for learning with programming performance $r = 0.512$, $p < 0.01$ and $r = 0.567$, $p < 0.01$ respectively. A regression model using these factors accounted for 60% of the variance in results
Byrnes and Lyons (2001)	BASIC	110	Overall result on 'Programming and Logical Methods' for first year Humanities students (70% written examination and 30% weekly assignments)	Significant correlations found for: LC mathematics result ($r = 0.353$, $p < .01$), LC science result ($r = 0.572$, $p < .01$) with programming performance

(*continued*)

Table 1. (*Continued*)

Researchers	Language	n	Reference criterion	Significant predictors
Cantwell-Wilson and Shrock (2001)	C++	105	Mid-term course grade on 'CS202 Introduction to Computer Science' for introductory computer programming students	A regression model using comfort-level, mathematics background and attribution of success/failure to luck accounted for 44% of the variance in mid-term results. Secondary analysis indicated that the number of hours playing computer games prior to the course had a negative effect on grade while experience of a prior formal programming class had a positive effect
Evans and Simkin (1989)	BASIC	117	Six outcome variables on an entry-level business computer class (homework score, 2 BASIC exams, 2 mid-term exams and a final examination)	Various predictors dependent upon outcome variable found (for example, number of high school mathematics courses, prior BASIC experience, hours playing video or computer games). Regression modelling accounted for at most 23% of the variance on each outcome variable
Gibbs (2000)	BASIC	50	Two achievement tests on a first programming course of computer science students. The first test was on designing and the second test was on coding	Within a constructivist learning environment, cognitive style was not found to influence programming achievement (on either tests)
Goold and Rimmer (2000)	C	39	Result on an introductory programming concepts course for students enrolled in a computer-related degree (60% examination and 40% assignments)	A regression model, composed of (in order) dislike for programming, gender, average on other modules and raw secondary score accounted for 43% of the variance in scores
Hagan and Markham (2000)	Java	97	Five outcome variables on a first programming course in an undergraduate computing degree (two tests 30%, two assignments 30% and examination 40%)	Significant difference found between the performance of students with and without prior programming experience on all assessments except the final examination

(*continued*)

Table 1. (*Continued*)

Researchers	Language	n	Reference criterion	Significant predictors
				Further analysis indicated that the more programming languages a student knew prior to taking the course, the higher the performance. They also found a significant difference between the confidence level of students with and without prior experience on their expectation to pass the class ($F = 4.558$, $p < 0.05$)
Holden and Weeden (2004)	Java	159	Three in-class exams on the first course of an 'Introduction to Programming' sequence	Prior experience (independent of language) is an advantage in the first course in a programming sequence but not in later courses
Honour-Werth (1986)	Pascal	58	Student grade on 'Computer Science I' (first class in the Association for Computing Machinery (ACM) Curriculum). Grade included scores on homework assignments 30%, examinations 40%, quizzes 10% and a programming project 20%	Significant correlations found for: high school mathematics ($r = 0.252$, $p < .05$), hours working at a part-time job ($r = 0.203$, $p < .1$), Piagetian intellectual development ($r = 0.232$, $p < .05$) and cognitive style ($r = 0.317$, $p < .01$) with performance
Hostetler (1983)	Fortran	79	Overall grade on an 'Introduction to Computers and Their Application to Business'. The grade consists of scores on programming assignments, two one-hour examinations and a three-hour final examination	A regression model, using diagramming and reasoning score on the Computer Programmer Aptitude Battery (CPAB), Grade Point Average (GPA), mathematics background and a personality factor (sober or happy-go-lucky trait), correctly classified 61 of 79 students (77.2%) into high and low aptitude groups
Konvalina et al. (1983)	Basic	382	Final examination on an 'Introduction to Computer Science' course based on CS1 in Curriculum '78	Found significant differences between the mathematics background of students who withdrew from the course and students who

(*continued*)

Table 1. (*Continued*)

Researchers	Language	n	Reference criterion	Significant predictors
				completed the course. Students who completed the course had significantly more mathematical background before taking the course
Kurtz (1980)	Fortran	23	Final grade on an 'Introduction to Programming' course for computer science majors and non-majors. Grade composed of scores on programs 55%, quizzes 10%, mid-term 11.7% and final examination 23.3%	Level of formal reasoning is a strong predictor of performance (specifically poor and outstanding performance), accounted for 63% of the variance in grades
Leeper and Silver (1982)	Not specified	92	Grade on an Introductory Programming Course	Regression model accounted for 26% of the variance using number of high school English, mathematics, science and foreign language units completed, Scholastic Assessment Test (SAT) verbal and SAT mathematics score and high school rank. Strongest correlations found for SAT verbal and SAT mathematics score
Mayer et al. (1986)	Basic	57	Basic examination score	Regression model accounted for 50% of the variance using two problem translation skills: word problem translation and word problem solution and a procedure comprehension skill: following directions
Newsted (1975)	Fortran	131	Grade on an Introductory Programming Course	Regression model using perceived ability, college GPA and time spent working on the course accounted for 41% of the variance. The first two variables were found to be positively related to performance but time spent working on the course was found to be negatively related

(*continued*)

Table 1. (*Continued*)

Researchers	Language	n	Reference criterion	Significant predictors
Ramalingam et al. (2004)	C++	75	Final grade on an introductory programming course for CS majors and non-majors	Student mental models found to directly affect programming performance and strengthen self-efficacy. Mental model, self-efficacy and previous programming and computer experience accounts for 30% of the variance in final course grade
Rountree et al. (2002)	Java	472	Final mark on a first year programming course. The mark was composed of 30% bi-weekly programming assignments, 20% mid-semester examination and 50% final examination.	The strongest indicator of success was the grade a student expected to get on the course
Stein (2002)	Java	160	Performance on CSII, a programming course	Students who study calculus do at least as well as students who study discrete mathematics. Thus, some form of mathematical maturity is important for programming
Ventura (2005)	Java	499	Course grade on a graphical design-centric objects-first course. Grade based on weighted average of homework and quizzes (10%), lab average (40%) and exam (50%)	A regression model using a log of percent lab usage, comfort-level and SAT mathematics score accounted for 52.9% of the variance in course averages
Wiedenbeck (2005)	C++	120	Final grade of non-majors on an introductory programming course	A measure of previous computer and programming experience along with measures of self-efficacy and knowledge organization accounted for 30% of the variance in final grade. Previous experience is a strong predictor of pre-self-efficacy

In recent studies researchers have examined various psychological factors including students' perceived comfort-level when learning to program. Cantwell-Wilson and Shrock (2001) in a recent longitudinal study found that the most important predictor of students' performance on an introductory computer science course was comfort level, determined by the degree of anxiety a student felt about the course. Goold and Rimmer (2000) identified that 'dislike of programming' influences performance on an introductory programming course. Ramalingam et al. (2004) and Wiedenbeck (2005) identified a positive relationship between students' mental models of programming and self-efficacy for programming and performance and suggested that knowledge organization directly affected success and strengthened post-efficacy. In a recent study by Bergin and Reilly (2005a), it was found that a student's perception of his or her understanding of the module had the strongest correlation with programming performance. In addition, Rountree et al. (2002) found that the grade a student expected to achieve in an introductory module was the most important indicator of performance. Ventura (2005) examined predictors of a graphical design-centric objects-first Java course and found that student effort (as measured by the number of hours spent using the labs) and comfort level were the strongest (positive) predictors of success. Newsted (1975) in a study of introductory Fortran, students found that two of the most important predictors of performance were perceived ability and time spent working on the course and working with other students. While perceived ability was found to be positively related to performance, the number of hours spent working on the course was negatively related to performance.

The previous sections have outlined the main body of research on factors that influence introductory programming success. Other factors have also been considered, albeit often in one-off studies that do not fall into the previous categories. In particular, two factors have been investigated in several studies and have been found to relate to programming performance. The factors are (1) the number of hours a student spends playing computer games and (2) the number of hours spent working at a part-time job. Cantwell-Wilson and Shrock (2001) found that the number of hours students played computer games was negatively related to performance on an introductory computer science course. Similarly, Evans and Simkin (1989) found the number of hours students spent playing electronic games (both video and computer games were studied) had a negative relationship with performance on an introductory Basic course. They also found that the number of hours a student spent working at a part-time job negatively relates to performance. Honour-Werth (1986) found a significant correlation between hours working at a part-time job, $r = 0.203$, $p < .1$, and performance on an introductory computer science course using Pascal.

Although numerous studies on factors that influence programming performance have been carried out, further predictors are still required. Recently, self-regulated learning (SRL) has become an important topic in education and psychology. Zimmerman (1986) defines SRL as the degree to which learners are metacognitively, motivationally and behaviourally active participants in their own academic learning. Furthermore, Pintrich (1990), and Zimmerman and Martinez-Pons (1990) advocate

that a complete model of SRL should incorporate cognitive and metacognitive strategies, referred to as a 'skill' component, and motivational components, referred to as 'will' components.

A considerable number of studies, including Pajares et al. (2000), Pokay and Blumenfeld (1990), Pintrich (1990) and Zimmerman and Martinez-Pons (1990), have consistently found a significant positive correlation between academic achievement and self-regulated learning among elementary, high school, and college students. With regard to programming performance, a recent study by Bergin and Reilly (2005c) found that students who perform well in programming use more meta-cognitive and resource management strategies than lower performing students. Furthermore, high levels of intrinsic motivation and task value are also associated with performing well in programming. A regression model based on cognitive, metacognitive and resource management strategies was able to account for 45% of the variance in student results.

While a considerable amount of research has been carried out on factors that affect programming performance, our interest is on factors that can be determined *early* in the academic year. Given this, a study to determine early identifiable factors that influence programming performance was conducted. The study builds upon the findings of previous studies and further investigates the usefulness of SRL as a predictor of performance.

3. Research Design

This section documents the methodology used in this study. A description of the participants, the instruments and the *a priori* procedures carried out on the data is provided.

3.1. Participants

The study was carried out at four third-level institutions (post-high school) in the Republic of Ireland in the academic year 2004–2005. In total, 123 students enrolled in a first-year introductory programming module voluntarily participated in this study. The sample was composed of 77.5% male students and 22.5% female students. Ninety-one percent of the sample had just completed second-level education, while 9% were mature students. Eighty-five percent of the sample had completed their second-level education in the Republic of Ireland while 15% had completed their second-level education abroad.

The four institutions involved in the study are referred to in this paper as Institute A, Institute B, Institute C and Institute D. The institutes were quite different in that one was a university, two were institutes of technology and one was a college of further education. Typically, entry requirements for Institute A would be higher than all the other institutes, while requirements for Institute B and C would be similar and higher than those of Institute D. The overall aim of each module was to provide students with introductory programming skills and the contents of each module were

highly similar. Students in Ireland do not study programming for national examination in secondary school (i.e. high school). An outline of the assessment structure for each module and a content overview is provided in Table 2. The measure of performance reported upon in this paper is the overall module mark.

At institutes B, C, and D, 85%, 70%, and 95% of the total student population agreed to participate respectively, while at Institute A 42% agreed to participate. Statistical tests were carried out to determine sample representativeness and are discussed in Section 3.3. Participants were provided with an information sheet about the research and signed a consent form agreeing to participate. Permission for the research activities was also granted by each of the participating institutions.

3.2. Instruments

Four instruments were used to collect data: a background questionnaire, a programming self-esteem questionnaire, a self-efficacy questionnaire and a motivation and learning strategies questionnaire. Data were collected in two study administrations. In the first administration all surveys (background questionnaire, Programming Self-esteem questionnaire, and the shortened Computer Programming Self-efficacy scale), except the Motivated Strategies for Learning Questionnaire (MSLQ), were administered. The first administration was carried out early in the programming module

Table 2. Module Overview

Institute	Language	Concepts covered*	Assessment structure
Institute A	Java	Variable types, selection statements, iteration, recursion, arrays, methods, sorting, searching, classes and objects	30% continuous assessment, 70% final examination
Institute B	Java	Variable types, selection statements, iteration, methods, classes and objects, introduction to applets	50% continuous assessment, 50% final examination
Institute C	Pascal	Variable types, selection statements, iteration, arrays, searching, sorting, linked lists and pointers	40% continuous assessment, 20% practical examination, 40% final examination
Institute D	VB	Variable types, selection, iteration, arrays, methods, classes and objects	100% project
Institute D	Java	Variable types, selection, iteration, arrays, methods, classes and objects	$2 \times 30\%$ assignments, 40% theory examination

*Not in order.

(when the students had completed very early programming concepts – typically variable types, selection statements and sometimes iteration) while the second administration was completed when they were on average one-third of the way through the material (typically around week 8 of a 24-week module). It was the intention that both administrations would be completed closer together but this was not possible due to timetabling and other constraints. The total time to complete all of the instruments varied between one hour and one hour 15 minutes at the participating institutions. A brief description of each of the instruments is provided next.

The background questionnaire collected data on a number of items including previous academic information, for example Leaving Certificate (LC) mathematics grade and highest LC science grade; prior programming and non-programming computer experience; comfort level on the module, and several miscellaneous items, including, number of hours playing games before and during the module, likely number of hours spent studying for the module and number of hours per week working at a (part-time) job. The questions on comfort level were based on a set of questions used in a study by Cantwell-Wilson and Shrock (2001). The questions examined a student's perception of his or her level of understanding compared to the rest of the class, his or her ease at asking and answering programming questions, his or her general understanding of programming concepts and his or her ability to design and complete assignments. The questions are referred to as 'Cantwell-Wilson and Shrock (2001) comfort-level questions' in this paper.

The Rosenberg Self-esteem (RSE) questionnaire (Rosenberg (1965)) was adapted to apply to programming self-esteem. The RSE scale is perhaps the most widely used self-esteem measure in social science research. The scale consists of ten questions and has been shown to have generally high inter-item reliability. Each of the questions were re-worded to relate to programming self-esteem and not to self-esteem directly. For example, the first question was changed from 'On the whole, I am satisfied with myself' to 'On the whole, I am satisfied with my programming progress'. In this paper the modified RSE questionnaire is referred to as the Programming Self-esteem questionnaire.

The Computer Programming Self-efficacy scale was designed by Ramalingham and Wiedenbeck (1998) and consists of 33 items which ask students to judge their capabilities in a wide range of programming tasks and situations. As this instrument was administered when students had very limited experience of the module material, a shortened version of this scale using only seven questions was used.

In this study we used the model of self-regulated learning developed by Pintrich and his colleagues, as outlined in Pintrich (1991a). This model stresses the learners' use of cognitive strategies and self-regulatory strategies, self-efficacy beliefs (individuals' judgements of their capabilities to perform a task), task value beliefs (the importance of, interest in and value associated with a task) and goal orientation (intrinsic and extrinsic goal orientation), (further details provided in Pintrich (1999)). The MSLQ (Pintrich (1991b)), a self-report instrument designed to measure students' motivation and self-regulated learning in classroom contexts, was used to measure this model. The MSLQ is composed of two sections: a motivation section

and a learning strategies section. To examine components of SRL the following scales were employed:

- *Goal orientation strategies*: intrinsic goal orientation scale and extrinsic goal orientation scale.
- Task-value scale.
- *Cognitive strategies*: rehearsal strategies scale, elaboration strategies scale and organization strategies scale.
- *Meta-cognitive strategies*: planning, monitoring and regulating strategies scale.
- Self-efficacy for learning and performance scale.

3.3. Data Pre-processing

While an in-depth discussion on data pre-processing carried out in this study is not appropriate in this paper, it is important to note that a number of *a priori* procedures were implemented to prepare the data for analysis. The procedures included (1) data screening, (2) testing the representativeness of the sample, (3) tests of unidimensionality, (4) dimensionality reduction and (5) tests to ensure the underlying assumptions of the statistical technique of choice, logistic regression, are satisfied. A detailed description of pre-processing can be found in Bergin and Reilly (2006). A few points from this process are worth briefly noting.

Data screening required the examination of encoded data to ensure that it was free of coding errors. Maximum and minimum frequency values were inspected to check that no out-of-bounds entries existed. As a more rigorous measure, each encoded item was inspected along with totals, by an independent witness and the author, to ensure that all data have been satisfactorily entered and computed.

An *a priori* analysis was carried out to verify no significant differences that existed between the mean overall module results of the students who participated in the study and the students who did not participate in the study. A t-test confirmed that no significant differences existed between the mean results of students who participated in the first administration of the study and the relevant student population at each institute. However, statistical differences were found between the students who participated in the motivation section ($t(67) = 6.451$, $p = 0.001$) and the learning strategies section ($t(56) = 7.1$, $p = 0.001$) of the MSLQ at institute A and this will have to be taken into account in the analysis. The cause of this statistical difference was a considerably reduced sample size on the second administration at institute A. This finding indicates that results involving the MSLQ may not be generalizable. To alleviate this problem, two separate investigations were carried out, the first using data gathered in the first administration and the second on the data gathered in both administrations. Interpretation of the second investigation will need to take into account the reduced sample size and the lack of sample representativeness.

Where multiple indicator variables were used to measure a construct, tests of unidimensionality were performed. Cronbach's alphas for each of the MSLQ subscales and the subsequent values calculated in this study are given in Table 3. In each

Table 3. Reliability analysis using Cronbach alpha measure for MSLQ scales as given by Pintrich et al. and as found in this study

Scale	Pintrich (1991b)	Study values
Intrinsic goal orientation scale	.74	.75
Extrinsic goal orientation scale	.62	.56
Task value scale	.90	.85
Self-efficacy for learning and performance	.93	.95
Rehearsal scale	.69	.73
Elaboration scale	.76	.58
Organization scale	.64	.63
Planning, monitoring and regulating scale	.79	.83

instance, the alpha values were found to be high. Test of reliability for the Cantwell–Wilson and Shrock comfort-level questions was 0.80 and for the shortened Computer Programming Self-efficacy scale was 0.95. Typically, Cronbach's alphas for the Rosenberg Self-esteem scale are in the range of 0.82 to 0.88, (Rosenberg (1965)) and for the modified Programming Self-esteem scale used in this study the alpha value was 0.91.

Principal components analysis (PCA) was performed on each of the multiple-item-based instruments and components that satisfied the Kaiser criterion (all components with eigenvalues greater than 1.0) were retained for future modelling. This resulted in one component for programming self-esteem, one component for self-efficacy and three components for the comfort-level questions.

The classification technique used was logistic regression. This technique makes no assumptions about the distributions of predictor variables, that is the predictors do not need to be normally distributed, linearly related or of equal variance within each group and can be a mix of continuous, discrete and dichotomous variables. The model produced by logistic regression is nonlinear and is denoted by

$$P_i = \frac{e^{b_O + b_i X_i}}{1 + e^{b_O + b_i X_i}}. \tag{1}$$

Like most statistical techniques, logistic regression does make some assumptions. A considerable number of procedures was put in place to ensure that the underlying assumptions were satisfied, including removal of outliers, procedures to reduce the likelihood of overfitting the data, and tests to ensure the absence of multicollinearity.

4. Results

Two investigations were carried out. In the first investigation all the factors from the first administration of the study were included, while in the second investigation all of the factors from both administrations were considered. In total, over 40 models were

developed with various degrees of freedom. All models were generated using 10-fold stratified cross-validation. With this procedure, data are randomly split into ten parts, with each part representing the same proportion of each class. Each part is held out in turn and the learning scheme is trained on the remaining nine parts, then the error rate is calculated on the holdout set. In total the procedure is executed ten times on different training sets and the results are averaged over all of the testing datasets.

4.1. Investigation 1

Numerous logistic regression models were developed using the variables from the first administration of the study. The most significant predictor set that emerged (Predictor Set 1) had three predictor variables, LC mathematics score (LCMATHEMATICS), number of hours spent playing games while taking the module (WHILEGAMES) and student values from PCA on the Programming Self-esteem scale (PROGSELFEST) (calculated as the sum of the first component score times the standard student response to each item in the scale). From a total of 123 cases, 102 were included in the model. The percentage of students accurately classified was significant at 80%. LCMATHEMATICS and PROGSELFEST were found to have a positive relationship with performance, while WHILEGAMES was found to have a negative effect (the more hours a student spent playing games the lower their performance on the module and vice versa). Table 4 outlines the high performance of this model. Although a considerable number of other models was developed, no superior predictor set was found. A second predictor set (Predictor Set 2) did emerge with marginally higher prediction accuracy 82% but with a reduced sample size $n = 82$. The predictor set included the three predictor variables from the first model and the number of hours students were likely to spend studying module material per week (LIKELYHOURS) and measures of performance are provided in Table 4.

In logistic regression, probabilities are used to determine to which class an instance (student) belongs. In general, the cut-off value is 0.5. Thus, we are not restricted to treating our outcome as dichotomous (weak or strong) but can also classify performance using the classification probabilities. This is an important benefit as it allows borderline students to be identified, that is, students who are clearly not very strong or very weak, for example students with a classification probability between 0.35 and, say, 0.65. For example, using the attributes described in Predictor Set 1, a

Table 4. Percentage of students correctly classified as weak or strong, as well as overall classification accuracy achieved by the logistic regression models

Model	Weak students correctly classified	Strong students correctly classified	Overall correctly classified
Predictor Set 1	87%	69%	80%
Predictor Set 2	.82%	.81%	.82%
Predictor Set 3	.96%	.93%	.95%

classification model can be derived using all 102 students as training data (for illustration purposes it is simpler to consider a single training set than 10-fold cross validation). Twenty students are misclassified. However, analysis of the classification probabilities indicates that ten of the misclassified students have a classification probability between 0.35 and 0.65 and thus form a borderline group of students. Assuming the objective is to assist weaker students, students in this borderline group should also be monitored. Of the remaining ten students, three are classified as strong but are actually weak and seven are classified as weak who are actually strong. If the objective is to assist weaker students in order to improve retention, it could be argued that the only significant error is the three students classified as strong who are weak.

4.2. Investigation 2

This investigation considered the complete set of attributes examined in the study. A significant predictor set emerged (Predictor Set 3) but one needs to be cautious interpreting the results due to the reduced sample size and the differences noted earlier between the sample group and population at Institute A and Institute C. The predictor set included LCMATHEMATICS, WHILEGAMES, LIKELYHOURS and the self-efficacy for learning and performance scale from the MSLQ (MSLQSELFEFF). Ninety-five percent of students ($n = 58$) were classified correctly and the model had high performance measures, as outlined in Table 4.

5. Discussion

In this section, a review of the instruments used in this study is provided and a discussion on the models developed is presented.

5.1. Review of the Instruments

While our Programming Self-esteem scale proved useful in our classification model, the other two comfort-level measures, the Cantwell-Wilson and Shrock (2001) comfort-level questions and the shortened Computer Programming Self-efficacy scale did not add any further value to the models. The Cantwell-Wilson and Shrock (2001) comfort-level questions, however, resulted in a slightly poorer classification model when used with LC mathematics and game playing (omitting PROGSELFEST). This suggests that it is measuring the same phenomena as Programming Self-esteem. However, the Programming Self-esteem scale is a superior measure. Given that we are trying to capture attributes at a very early stage in the programming course, only seven questions asking students to judge their ability at specific programming tasks, from the Computer Programming Self-efficacy scale, could be administered. Clearly, this shortened version is not sufficient to capture self-efficacy.

Analysis of the SRL measures using independent t-tests revealed that weaker students had lower intrinsic motivation than stronger students ($t(77) = -3.298$, $p = 0.001$). In addition, weaker students used less meta-cognitive strategies

(specifically, planning, monitoring and regulating) than stronger students ($t(79) = -4.566$, $p = 0.001$). While, use of meta-cognitive strategies and intrinsic motivation level do not increase the accuracy of the models (perhaps because the information they provide is already captured in the model), this information is useful to educators who are seeking to help students learn programming.

Many of the items in the background survey did not prove useful in the development of the prediction models. These items included, prior programming experience, encouragement from others to study programming, preference to work alone or in a group when solving problems and number of hours using application software, emailing or surfing the web before and during the early stages of the course. Previous studies have reported conflicting results on prior programming experience. However, in the Republic of Ireland students do not study programming in secondary school at examination level and this may account for why it is not an indicator of performance in this study. Our findings on encouragement from others to study programming, preference to work alone or in a group when solving problems and number of hours using application software, emailing or surfing the web before and during the early stages of the course are in line with a longitudinal study carried out by Cantwell-Wilson and Shrock (2001).

5.2. Analysis of Predictor Set 1

The fact that LC mathematics is a useful predictor is of no great surprise given our literature review in Section 2. The Programming Self-esteem measure had never been used before, but it can be thought of as another measure of comfort level. Considered as such, the findings on its predictiveness are in line with previous research. The importance of computer game playing as a predictor of performance supports the results of Cantwell-Wilson and Shrock (2001) and Evans and Simkin (1989), who similarly found a negative relationship between game playing and programming performance.

Predictor Set 1 considers the largest number of students ($n = 102$). When a separate logistic regression model is developed for each institution, prediction accuracy remains high, with accuracies of 85% at Institute A, 96% at Institute B, 92% at Institute C and 71% at Institute D, respectively. The lower result at Institute D can be accounted for by the fact that only 44% students were included in the classification due to a large amount of missing data. In most cases, LC mathematics score was missing. A large proportion of the students at this college had obtained their second-level education abroad and consequently did not sit the LC mathematics examination. This is a problem for the current model. Although several substitution schemes were examined, none were found suitable. We intend to examine other substitution schemes to alleviate this problem and to investigate alternative classification techniques for handling missing data better.

The order of importance of the three variables changes at the different institutions, as illustrated in Table 5. It is interesting that the exact same ordering of predictors is found at both Institute B and C. Both of these institutes have similar admission

Table 5. Order of importance for attributes

	Institute A	Institute B	Institute C	Institute D
1st	PROGSELFEST	WHILEGAMES	WHILEGAMES	WHILEGAMES
2nd	LCMATHEMATICS	LCMATHEMATICS	LCMATHEMATICS	PROGSELFEST
3rd	WHILEGAMES	PROGSELFEST	PROGSELFEST	LCMATHEMATICS

requirements and are the same type of institute (institutes of technology). As such, students with similar academic backgrounds would attend each college and the model appears to adjust accordingly for this. Institute A is a university and, in general, would have a higher admissions requirement. Computer game playing at this institution is the least important predictor as opposed to the most important at the others. It could be interpreted that students at Institute A who, in general, would have higher entry results, work harder and are less likely to play games. However a one-way ANOVA test failed to reveal any statistical differences between computer game playing at each of the institutions. An ANOVA test revealed a statistical difference between the mean LC mathematics score at Institute A and each of the other institutes ($F(3,98) = 24.985$, $p < 0.001$). There was no difference in the mean score at institutes B, C and D. This could partially explain why LC mathematics is a more important factor at Institute A. No statistical differences were found between the programming self-esteem scores at the different institutions.

5.3. Predictor Set 2

Predictor Set 2 includes the likely number of hours a student will spend studying for the module. The inclusion of this attribute results in an improvement in the number of students classified as strong (from 69% to 81%), but with a slight decrease in the prediction performance of weak students (from 87% to 82%). An ANOVA test failed to reveal any significant differences between the likely hours studied by weak and strong students. It appears that knowing the number of hours a strong student will spend studying helps to classify strong students but the same is not true of weak students. This may be caused by weak students overestimating the number of hours they will spend studying, however further investigation is warranted.

5.4. Predictor Set 3

The measures of self-regulated learning were disappointing. The only measure that added to the classification model was the MSLQ self-efficacy scale. This is not surprising given the findings on the importance of self-efficacy in other studies. When this measure is included in the model, the Programming Self-esteem measure does not contribute any significant additional value. To determine whether the MSLQ self-efficacy scale was a superior measure than the Programming Self-esteem scale,

albeit with a considerably reduced sample size, $n = 58$, a further logistic regression model was developed. The model used the same student sample as the model developed using Predictor Set 3 ($n = 58$) but used the Programming Self-esteem measure instead of the MSLQ self-efficacy scale along with LCMATHEMATICS, WHILEGAMES, and LIKELYHOURS. The model resulted in poorer prediction accuracy with 88% of the students classified correctly. Given the problems of sample representativeness in this study, it is not possible to determine if the MSLQ self-efficacy scale measure is a truly superior measure than the Programming Self-esteem scale, however future studies should endeavour to verify this finding.

Further analysis on the SRL measures using independent t-tests revealed that weaker students had lower intrinsic motivation than stronger students. In addition, weaker students used fewer meta-cognitive strategies (specifically, planning, monitoring and regulating) than stronger students. This is in line with previous findings on SRL and programming performance, as outlined in Bergin and Reilly (2005c).

6. Epilogue

In the academic year 2005 – 2006 students enrolled on the introductory programming module at Institute A were asked to participate in an additional study to verify the effectiveness of the three factors and also to verify the suitability of logistic regression for predicting programming performance. Students were asked to answer questions based on the three factors identified in Predictor Set 1, that is their LC mathematics result, the number of hours spent playing computer games and the Programming Self-esteem scale. Twenty-one of the 22 students (95%) who completed the module participated in the study. The study was carried out when the students had completed three weeks of Java programming (variable types, selection statements and iteration).

The full set of students who participated in the main study outlined in this paper with no missing data, ($n = 102$), were used as training instances to develop a final logistic regression model. The model achieved an overall prediction accuracy of 81% (4 students were misclassified). The number of students correctly classified as weak was 80% (2 students misclassified) and the number of students correctly classified as strong was 82% (2 students misclassified). With regards to the two students who were predicted to be 'strong' programmers but were actually 'weak', the first student achieved an overall result of 54.97% and the cut-off value for weak was 55.5%. That is, had the student achieved 0.6% more they would have been correctly classified, increasing the overall accuracy measure to 86% and the sensitivity measure to 90%. The second student who was misclassified as 'strong' did not attend any lab or workshop sessions and attended less than 5% of the lectures in the second semester. Prior to their non-attendance the student had performed well in their class and lab exams.

This study further confirms the effectiveness of a logistic regression model using the three identified factors for predicting programming performance.

6.1. Conclusions

The work outlined in this paper makes four main contributions to the field. First, the research is based on a new study on identifying factors that influence success on introductory programming modules. Over 25 factors were examined at four different institutions. The study provided further evidence on the importance of mathematics, comfort level and game playing as predictors of programming performance. The study also examined numerous other factors and found that they failed to contribute further to the prediction model, for example prior programming experience, number of hours a student spends working at a part-time job, encouragement from others to study programming, preference to work alone or in a group when solving problems and number of hours using application software, emailing or surfing the web before and during the early stages of the course. Second, the study introduced the use of a new instrument, (the Programming Self-esteem scale), and found that it outperforms the Cantwell-Wilson and Shrock (2001) comfort-level questions and the shortened Computer-programming Self-efficacy scale as a measure for predicting programming performance. Third, an investigation on the usefulness of SRL for predicting performance was carried out. While, SRL was not found to contribute further to the model, except for the self-efficacy measure, its role in learning to program was identified and justifies further research in the area. Finally, the classification model outlined in investigation 1 and the subsequent analysis using classification probability is arguably one of the most successful prediction models to date. The use of 10-fold stratified cross-validation further confirms the generalizability of the findings.

Future work should seek to validate the second model. The MSLQ self-efficacy scale should be administered along side the other measures to see if it is useful when administered in the very early stages of the module. If it is found to be useful, then incorporating the measure with the number of hours a student is likely to study, their LC mathematics score, and the number of hours playing games could result in an even more effective model.

A future study examining the effectiveness of the three-factor model at predicting performance in other computer science topics, for example, discrete mathematics would be useful. The only modification required to the instruments would be to re-word the programming self-esteem measure to represent the new topic. Where students study computer science as part of a science or arts degree, it would also be useful to determine how well the model predicts performance on the associated science or arts modules.

With regards to the LC mathematics score, studies need to be carried out in other countries to see if performance on other mathematics tests can be used in the model instead. The development of a mathematical test that captures the aspects of mathematics that are most important in learning to program would be very useful rather than relying on responses on a test that all students may not have taken.

Acknowledgements

The authors express their gratitude to the Higher Education Authority (HEA) for partially funding this work. We also sincerely thank the students who participated and the staff who facilitated this study, in particular, Karel de Raeymaeker, Margaret Kinsella and Enda Dunican.

References

Austin, H.S. (1987). Predictors of Pascal programming achievement for community college students. *ACM SIGCSE Bulletin*, *19*(1), 161–164.

Barker, R.J., & Unger, E.A. (1983). A predictor for success in an introductory programming class based upon abstract reasoning development. *ACM SIGCSE Bulletin*, *15*(1), 154–158.

Bergin, S., & Reilly, R. (2005a). Programming: factors that influence success. *ACM SIGCSE Bulletin*, *37*(1), 411–415.

Bergin, S., & Reilly, R. (2005b). The influence of motivation and comfort-level on learning to program. *Proceedings of the 17th Workshop on Psychology of Programming,* PPIG '05, 293–304.

Bergin, S., & Reilly, R. (2005c). Examining the role of self-regulated learning on introductory programming performance. *Proceedings of the 2005 international Workshop on Computing Education Research,* ICER 2005, 81–86.

Bergin, S., & Reilly, R. (2006). Statistical and machine learning models to predict programming performance. PhD thesis.

Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. *ACM SIGCSE Bulleting*, *33*(3), 49–52.

Cantwell Wilson, B., & Shrock, S. (2001). Contributing to success in an introductory computer science course: a study of twelve factors. *ACM SIGCSE Bulletin 33*(1), 184–188.

Evans, G.E., & Simkin, M.G. (1989). What best predicts computer proficiency? *Communications of the ACM*, *32*(11), 1322–1327.

Gibbs, D.C. (2000), The effect of a constructivist learning environment for field-dependent/independent students on achievement in introductory computer programming. *ACM SIGCSE Bulletin*, *32*(1), 207–211.

Goold, A., & Rimmer R. (2000). Factors affecting performance in first-year computing. *ACM SIGCSE Bulletin*, *32*(2), 39–43.

Hagan, D., & Markham, S. (2000). Does it help to have some programming experience before beginning a computing degree program? *ACM SIGCSE Bulletin*, *32*(3), 25–28.

Honour-Werth, L. (1986). Predicting student performance in a beginning computer science class. *ACM SIGCSE Bulletin*, *18*(1), 138–143.

Holden E., & Weeden, E. (2004). The impact of prior experience in an information technology programming course sequence. *Proceedings of the 4th Conference on Information Technology Curriculum,* CITC4 '03, 41–46.

Hostetler, T.R. (1983). Predicting student success in an introductory programming course. *ACM SIGCSE Bulletin*, *15*(3), 40–43.

Konvalina, J., Wileman, S.A., & Stephens, L.G. (1983). Math proficiency: a key to success for computer science students. *Communications of the ACM*, *26*(5), 377–382.

Kurtz, B.L. (1980). Investigating the relationship between the development of abstract reasoning and performance in an introductory programming class. *ACM SIGCSE Bulletin*, *12*(1), 110–117.

Leeper, R.R., & Silver, J.L. (1982). Predicting the success in a first programming course. *ACM SIGCSE Bulletin*, *14*(1), 147–150.

Mayer, R.E., Dyck, J.L., & Vilberg, W. (1986). Learning to program and learning to think: what's the connection? *Communications of the ACM*, *29*(7), 605–610.

Newsted, P.R. (1975) Grade and ability predictions in an introductory programming course. *ACM SIGCSE Bulletin*, *7*(2), 87 – 91.

Pajares, F., Brinter, S., & Valiante, G.(2000). Relation between achievement goals and self-beliefs of middle school students in writing and science. *Contemporary Educational Psychology*, *25*(4), 406 – 422.

Pintrich, P., & DeGroot, E. (1990). Motivational and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology*, *82*(1), 33 – 40.

Pintrich, P., & Garcia, T. (1991a). Student goal orientation and self-regulation in the college classroom. *Advances in Motivation and Achievement*, *7*, 371 – 403.

Pintrich, P., Smith, D., Garcia, T., & McKeachie, W. (1991b). A manual for the use of the motivated strategies for learning questionnaire. Technical Report 91-B-004. The Regents of the University of Michigan.

Pintrich, P. (1999). The role of motivation in promoting and sustaining self-regulated learning. *International Journal of Educational Research*, *31*, 459 – 470.

Pokay, P., & Blumenfeld, P. (1990). Predicting achievement early and late in the semester: the role of motivation and learning strategies. *Journal of Educational Psychology*, *82*(1), 41 – 50.

Ramalingham, V., & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research*, *19*(4), 367 – 381.

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004). Self-efficacy and mental models in learning to program. *ACM SIGCSE Bulletin*, *36*(3), 171 – 175.

Rosenberg, M. (1965). *Society and the adolescent self image.* Princeton, NJ: Princeton University Press.

Rountree, N., Rountree, J., & Robins, A. (2002). Predictors of success and failure in a CS1 course. *ACM SIGCSE Bulletin*, *34*(4), 121 – 124.

Stein, M.V. (2002) Mathematical preparation as a basis for success in CS-II. *Journal of Computing in Small Colleges*, *17*(4), 28 – 38.

Ventura, P.R. (2005). Identifying predictors of success for an objects-first CS1. *Computer Science Education*, *15*(3), 223 – 243.

Wiedenbeck, S. (2005). Factors affecting the success of non-majors in learning to program. *Proceedings of the 2005 International Workshop on Computing Education Research*, ICER 2005, 13 – 24.

Zimmerman, B. (1986). Becoming a self-regulated learner: which are the key sub-processes? *Contemporary Educational Psychology*, *11*(4), 307 – 313.

Zimmerman, B., & Martinez-Pons, M. (1990). Student differences in self-regulated learning: relating grade, sex and giftedness to self-efficacy and strategy use. *Journal of Educational Psychology*, *82*(1), 51 – 59.